RDC

ROCKWELL DESIGN CENTER

LOW COST EMULATOR (LCE)

SYSTEM USER'S MANUAL

Rockwell International

Document No. 29655N19
Order No. RDC19
December 1984

Ron Collins - 714-833-4746 -4522
Xio Beach - 714 - 833-4509
Sales
Gary Scott 4655

RDC

ROCKWELL DESIGN CENTER

LOW COST EMULATOR [LCE]

SYSTEM USER'S MANUAL

Rockwell International
Semiconductor Products Division

©Rockwell International Corporation 1984
All Rights Reserved
Printed in U.S.A.

Document No. 29655N19
Order No. RDC19
December 1984

NOTICE

Rockwell International does not assume any liability arising out of the application or use of any products, circuit or software described herein, neither does it convey any license under its patent rights nor the patent rights of others. Rockwell International further reserves the right to make changes in any products described herein without notice. This document is subject to change without notice.

WARNING

This equipment generates, uses and can radiate radio frequency energy and if not installed and used in accordance with the instruction manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to subject J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

INFORMATION TO USER

If this equipment does cause interference to radio or television reception which can be determined by turning the equipment on and off, the user is encouraged to try to correct the interference by one or more of the following measures:

- reorient the receiving antenna
- relocate the computer with respect to the receiver
- move the computer away from the receiver
- plug the computer into a different outlet so that the computer and receiver are on different circuits.

If necessary, the user should consult the dealer or an experienced radio/ television technician for additional suggestions. The user may find the following booklet prepared by the Federal Communications Commission helpful:

"How to Identify and Resolve Radio and TV Interference Problems"

This booklet is available from the U.S. Government Printing Office, Washington, D.C. 20402, Stock Number 004-000-0345-4.

TABLE OF CONTENTS

Section Page

SECTION 1
INTRODUCTION

1.1 LCE Software Features 1-1
1.2 LCE Options 1-3
1.2.1 LCE R6500/* Personality Set. 1-3
1.2.2 LCE R6502-R65C02 Personality Set 1-5
1.2.3 Target RAM Modules 1-6
1.2.4 R6500 Cross Assembler. 1-6
1.3 Related Documents 1-7

SECTION 2
LCE COMMAND STRUCTURE

2.1 Default Modes 2-1
2.2 Primary Menu 2-4
2.3 General System Characteristics. 2-5
2.4 Direct Function Execution 2-6

SECTION 3
SYSGEN COMMAND MODES

3.1 PORT. 3-2
3.1.1 BAUD 3-3
3.1.2 WORD LEN 3-3
3.1.3 DELAY. 3-4
3.1.4 FORMAT 3-4
3.1.5 START CHAR 3-5
3.1.6 STOP CHAR. 3-5
3.1.7 SYSTEM 3-5
3.2 TARGET. 3-6
3.2.1 CPU/MPU Selection. 3-6
3.2.2 Prototype Memory Selection 3-7
3.3 PROTECT 3-8
3.3.1 PROTECT Memory Area. 3-8
3.3.2 UNPROTECT Memory Area. 3-9
3.3.3 CLEARALL Memory Areas. 3-10
3.3.4 DISPLAY Protected Memory Areas 3-10

TABLE OF CONTENTS (CONTINUED)

Section Page

3.5 BEEPER. 3-10
3.6 DISPLAY 3-11
3.7 EXIT. 3-11

SECTION 4
DEBUG FUNCTIONS

4.1 DISPLAY/ALTER MEMORY. 4-3
4.1.1 DUMP 4-3
4.1.2 MODIFY 4-5
4.2 ASSEMBLY 4-6
4.2.1 ASSEMBLER. 4-6
4.2.2 DISASSM 4-9
4.3 LOAD/SAVE MEMORY. 4-10
4.3.1 LOAD 4-10
4.3.2 SAVE 4-11
4.3.3 VERIFY 4-12
4.4 EXECUTION 4-13
4.4.1 STEP 4-13
4.4.2 RUN. 4-15
4.4.3 SNAPSHOT 4-16
4.4.4 MATCH. 4-17
4.5 RESET 4-18
4.6 DISPLAY/ALTER REGISTERS 4-18
4.6.1 PC (Program Counter) 4-19
4.6.2 A (A Register) 4-19
4.6.3 X (X Register) 4-19
4.6.4 Y (Y Register) 4-19
4.6.5 PS (Processor Status). 4-20
4.6.6 SP (Stack Pointer) 4-20
4.6.7 ALL. 4-20
4.6.8 KEEP 4-20
4.7 BREAK 4-21
4.7.1 BP1 Through BP5. 4-21
4.7.2 CLRALL 4-26

TABLE OF CONTENTS (CONTINUED)

Section Page

SECTION 5
SYMBOL MENU FUNCTIONS

5.1 LIST. 5-3
5.2 CREATE. 5-3
5.3 SYMBOL. 5-4
5.4 OBJECT. 5-4
5.5 LOADSYM 5-5
5.6 SAVESYM 5-5
5.7 CLRSYM. 5-6
5.8 SHOW. 5-6

SECTION 6
TERMINAL MODE

6.1 TERMINAL. 6-1

SECTION 7
MISCELLANEOUS FUNCTIONS

7.1 SPEED 7-1
7.2 PROM 7-1
7.3 TRACE 7-1

APPENDICES

Appendix		Page
A	LCE Memory Map	A-1
B	LCE Jumper and Switch Settings	B-1
	B.1 SBC Module	B-1
	B.2 ACIA Module	B-2
	B.3 UM Module	B-3
	B.4 PMC Module	B-4
C	LCE Command Summary	C-1
	C.1 SYSGEN Menu Command Summary	C-3
	C.2 MEMORY Menu Command Summary	C-8
	C.3 EMULATE Menu Command Summary	C-9
	C.4 SYMBOL Menu Command Summary	C-12
	C.5 TERMINAL Command	C-14
	C.6 PROM Commands	C-14
	C.7 TRACE Commands	C-14
	C.8 SPEED Command	C-14
D	Personality Set Error Messages	D-1
E	Program Assembly and Download on the IBM Personal Computer	E-1
F	Program Assembly and Download on the Intel Development System	F-1

LIST OF FIGURES

Figure		Page
1-1	Rockwell Low Cost Emulator (LCE)	1-2
1-2	LCE System Configuration Guide	1-4
2-1	Default Entry Display for LCE Power-On and RESET	2-3

LIST OF TABLES

Table		Page
2-1	LCE Default Switch Definitions	2-2

SECTION 1

INTRODUCTION

The Rockwell Low Cost Emulator (LCE) is a low-cost microcomputer development system for the R6500 microcomputer family. The LCE is packaged in an attractive case for bench-top or desk-top operation. The LCE is capable of debugging computer programs written for execution by the R6500/* one-chip microcomputers as well as R6500 and R65C00 microprocessors through the use of optional personality sets. R6500 programs are normally cross assembled on a host development system or personal computer and then the object code is downloaded to the LCE over a communications channel. Debugging can then be done on the LCE using an optional R6500/* or R6502-R65C02 Personality Set. All LCE system software is ROM-resident for ease of use. Figure 1-1 shows the LCE connected to a user-provided terminal and to an optional personality set.

This manual contains a complete description of the software function available on the LCE and the procedures to use these functions. These functions include all the system utilities and the debug software that operate in conjunction with an installed personality set.

1.1 LCE SOFTWARE FEATURES

- Self-contained, ROM-based operating system including system test
- Softkey function access to menu-driven operational modes
- SYSGEN configurable to user environment
- Software allows choice of any baud rate, word length or parity options with full handshake signal support from the R6551 ACIA device
- Debugging capability for optional personality sets for R6500/* one-chip microcomputers and R65XX-R65CXX microprocessors
- R6500 assembler with all instruction subsets
- Five hardware breakpoints, 32-bits wide (16 address, 8 data and 6 control), with "don't care" mask for each bit
- Configurable as a satellite for any host system including the Rockwell SYSTEM 65

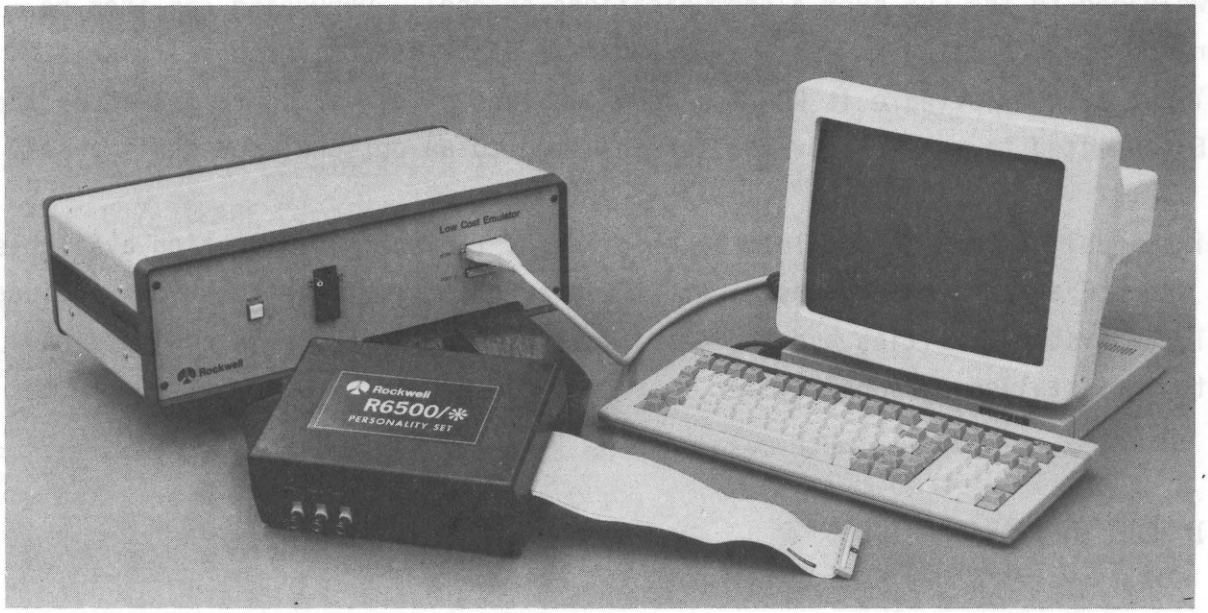


Figure 1-1. Rockwell Low Cost Emulator (LCE)

1.2 LCE OPTIONS

The LCE must always be used in conjunction with a personality set and, often, with target RAM modules. The personality set options allow the user to create the optimum development environment for the device to be emulated and still allow the the target device to change with minimum cost impact. In addition; user terminal, host computer or accessories can also be used as shown in the LCE system configuration guide (Figure 1-2). Refer to the LCE Hardware Installation Manual for additional information.

1.2.1 LCE R6500/* Personality Set

The LCE R6500/* Personality Set (part no. RDC-3X9) allows debugging of programs designed for execution by an R6500/* one-chip microcomputer. The option consists of two major components:

- The personality pod housing two emulator pod modules. These modules contain the microcomputer emulator device and the associated buffering and control logic. The personality pod is the link between a user's prototype hardware and the PMC module. The personality pod allows the user to test software in an isolated environment (i.e., personality pod disconnected from prototype circuit) or in-circuit (i.e., personality pod connected to prototype circuit).
- In-circuit emulation probe, the physical link between a user's prototype circuit and the personality pod. This probe is used during in-circuit emulation testing and is available in a 40-pin DIP or 64-pin QUIP configuration.

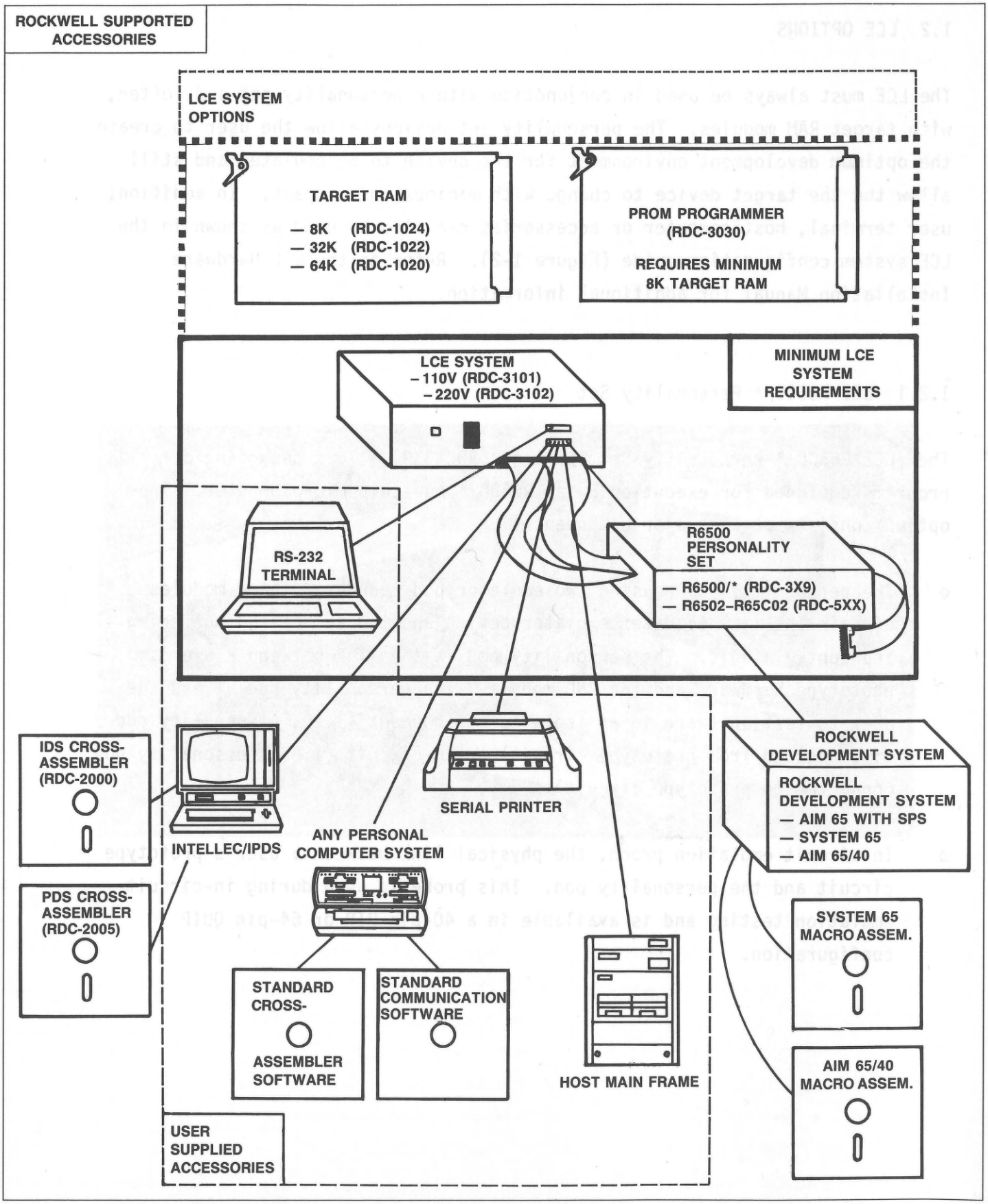


Figure 1-2. LCE System Configuration Guide

The R6500/* Personality Set includes model options to support emulation of the following devices:

R6500/11	R6501Q
R6500/12	R6541Q
R6500/15	R6500/41
R6500/16	R6500/42
R6511Q	R6500/43
R6500/13	R6500/1

For additional information, refer to the R6500/* Personality Set Hardware Installation Manual.

1.2.2 LCE R6502-R65C02 Personality Set

The LCE R6502-R65C02 Personality Set (part no. RDC-509) allows debugging of programs designed for execution by the R65XX and R65CXXX family of microprocessors. The option consists of two major components:

- The personality pod housing two emulator pod modules. These modules contain the microprocessor emulator device and associated buffering and control logic. The personality pod is the link between a user's prototype hardware and the PMC module. The personality pod allows the user to test software in an isolated environment (i.e., personality pod disconnected from prototype circuit) or in-circuit (i.e., personality pod connected to prototype circuit).

- In-circuit emulation probe, the physical link between a user's prototype circuit and the personality pod. This probe is used during in-circuit emulation testing and is available in a 28-pin DIP or 40-pin DIP configuration.

The R6502-R65C02 Personality Set directly supports emulation of the following devices:

R6502	R6507	R65C02
R6503	R6512	R65C102
R6504	R6513	R65C112
R6505	R6514	
R6506	R6515	

For additional information, refer to the R6502-R65C02 Personality Set Hardware Installation Manual.

1.2.3 Target RAM Modules

Any time a device is being emulated, using either the R6500/* or the R6502-R65C02 Personality Set, additional read/write memory is required in the LCE to hold object code that is being debugged. This "target" RAM is available in 8K-, 32K- or 64K-byte configurations (part numbers RDC-1020, RDC-1022 and RDC-1024, respectively). For additional information, refer to the Target RAM module Installation Manual.

1.2.4 R6500 Cross Assembler

Rockwell R6500 cross assemblers are available to assemble R6500, R6500/* and R65C00 microcomputer and microprocessor programs on the Intel Development System (part no. RDC-2000) or Intel Personal Development System (part no. RDC-2005). The cross assembler allows 6500 computer programs to be assembled and saved to disk. The object code can then be downloaded to the LCE for debugging and real-time emulation.

These disk-based, two-pass assemblers produce absolute R6500, R6500/* or R65C00 object code. They perform symbol definition, syntax checking, and generate object code, assembly/symbol table listings and symbol cross reference listing for effective program development. This absolute object code is directly executable on any 6500-based microprocessor or microcomputer,

e.g., R6500/1 One-Chip Microcomputer, R6502 NMOS Microprocessor or R65C02 CMOS Microprocessor. For additional information, refer to the R6500 Cross Assembler for Intel Development Systems User's Manual.

1.3 RELATED DOCUMENTS

The following is a list of related Rockwell documents which provide additional information about the LCE components and general reference.

Order No./Document No.	Title
Data Sheets	
RDC17/29655N17	RDC-310X Low Cost Emulator Data Sheet
RDC06/29655N06	RDC-3XX R6500/* Personality Set Data Sheet
RDC11/29655N11	RDC-50X R6502-R65C02 Personality Set Data Sheet
RDC18/29655N18	RDC-102X Target RAM Module Data Sheet
RDC09/29655N09	RDC-2000 R6500 Cross Assembler for Intel Development System Data Sheet
RDC23/29655N23	RDC-2005 R6500 Cross Assembler for Intel Personal Development System Data Sheet
Manuals	
201 /29650N31	R6500 Hardware Manual
202A /29650N30A	R6500 CPU Programming Manual
RDC20/29655N20	Low Cost Emulator (LCE) Hardware Installation Manual
RDC05/29655N05	R6500/* Personality Set for LCE and RDC Mainframe Hardware Installation Manual
RDC10/29655N10	R6502-R65C02 Personality Set for LCE and RDC Mainframe Hardware Installation Manual
RDC02/29655N02	RDC R6500 Cross Assembler for Intel Development System User's Manual
RDC16/29655N16	Target RAM Module Installation Manual

SECTION 2

LCE COMMAND STRUCTURE

2.1 DEFAULT MODES

When the LCE is powered up, or a COLD RESET is performed (i.e., the LCE RESET switch is pressed and released when Default Switch S1-3 is closed), the default SYSGEN parameters are assumed and SYSGEN is entered. The baud rate and word length can be selected by the LCE Default switch positions as summarized in Table 2-1. For a detailed description of the all Default Switch controlled options, refer to Section 2.4 of the LCE Hardware Installation Manual.

A typical default screen for power up (switches S1-1 to S1-8 set to OOXOOOCC) or COLD RESET (switches S1-1 to S1-8 set to OOCOOOCC) is shown in Figure 2-1a. This entry to SYSGEN allows the user to set up the desired operating parameters. After all parameters are correct (see Section 3), press the EXIT softkey to return to the PRIMARY menu.

Typical operation of the LCE would have S1-3 in WARM RESET position. This allows the reset to be used without disturbing user default parameter values or clearing the symbol table. The PRIMARY menu (Figure 2-1b) is displayed in this case (see Section 2-2).

NOTE

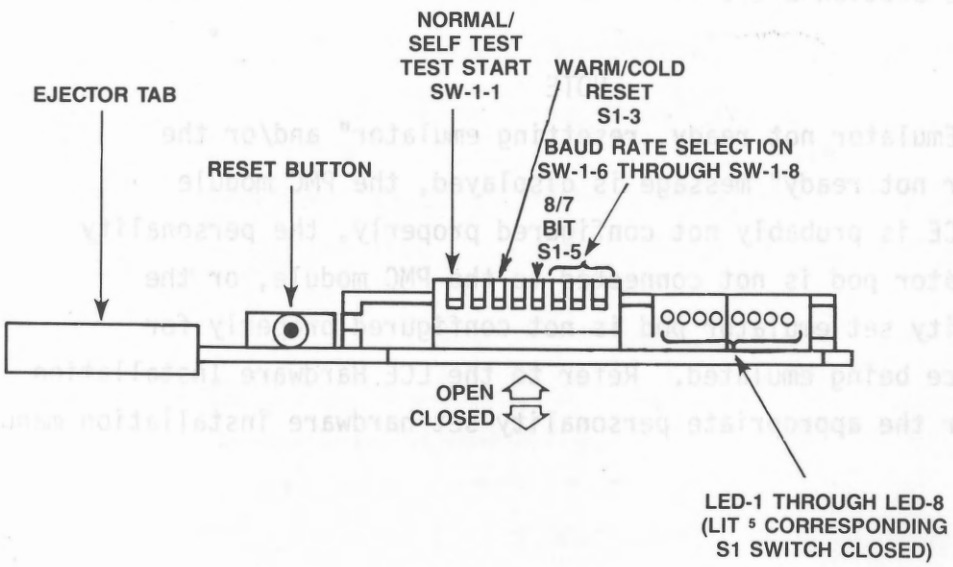
If the "Emulator not ready, resetting emulator" and/or the "Emulator not ready" message is displayed, the PMC module in the LCE is probably not configured properly, the personality set emulator pod is not connected to the PMC module, or the personality set emulator pod is not configured properly for the device being emulated. Refer to the LCE Hardware Installation Manual or the appropriate personality set hardware installation manual.

Table 2-1. LCE Default Switch Definitions

Switch	Function	Description
S1-1	Operating Mode	O = NORMAL Operation* C = SELF-TEST Operation
S1-2	Not Used	
S1-3	Reset Mode	O = WARM RESET* C = COLD RESET
S1-4	Not Used	
S1-5	Word Length	O = 8 Bits* C = 7 Bits
S1-6 - S1-8	Baud Rate	S1-6 S1-7 S1-8 Baud O O O 110 C O O 300 O C O 600 C C O 1200 O O C 2400 C O C 4800 O C C 9600* C C C 19200

Legend: O = OPEN C = CLOSED

*S1-1 through S1-8 factory shipped configuration = 000000CC



```
  Low Cost Emulator        Version 1.0
 Rockwell International   copyright 1984

Default switches :   9600 baud   8 bits    Warm Reset

Port 1:   9600 baud   8 bits     [SYSTEM PORT]   Beeper is OFF
LF delay is 00 ms   File format is ;    Start char is FF   Stop char is FF

Port 2:   9600 baud   8 bits
LF delay is 00 ms   File format is ;    Start char is FF   Stop char is FF

Target:     65XX is Selected
No Prototype Memory
No Protected Areas

*** SYSTEM OPTIONS ***

 8K of System RAM
Prom Programmer option is not present
Trace option is not present

 1:PORT 1  2:PORT 2  3:TARGET  4:PROTECT  5:BEEPER  6:DISPLAY  7:EXIT
```

a. LCE Power up or COLD RESET

```
  Low Cost Emulator        Version 1.0
 Rockwell International   copyright 1984
```

b. LCE WARM RESET

Figure 2-1. Default Entry Display for LCE Power-On and RESET

2.2 PRIMARY MENU

The PRIMARY menu is the top operating level for the LCE. Entry to this menu is on any WARM RESET, ESC from any menu level, exit from SYSGEN or TERMINAL modes, and typical movement between functions using the "backup menu" function (commanded by pressing the 0 or - key, refer to Section 2.3). The PRIMARY commands are selected from the menu using the softkeys:

1:SYSGEN 2:MEMORY 3:EMULATE 4:SYMBOL 5:TERMINAL 6:PROM 7:TRACE 8:SPEED

The PRIMARY functions are, in summary:

SYSGEN - Allows the user to set up the system parameters for all I/O and Personality Module Controller (PMC) functions that do not change during a debug session. This softkey menu is displayed first when the system is turned on or following a COLD RESET (refer to Section 3).

MEMORY - Displays and allows altering of the contents of target and protype memory during a debug session (refer to Section 4).

EMULATE - Provides all of the functions for directly controlling run-time characteristics and emulation (refer to Section 4).

SYMBOL - Contains the functions which control the generation and use of the symbol tables as it relates to the MEMORY functions (refer to Section 5).

TERMINAL - This mode echos every character typed on the terminal to the host. The character sent to the host is then sent back to the terminal and displayed (refer to Section 6).

PROM - Selects the functions for use with the optional PROM Programmer module (refer to Section 7).

TRACE - Selects the functions for use with the optional TRACE module (refer to Section 7).

SPEED - Defines the speed (0 is the fastest, 9 is the slowest) at which data is output to the console during DUMP and DISASSM functions (refer to 7).

2.3 GENERAL SYSTEM CHARACTERISTICS

There are several general characteristics of the LCE which pertain to all of the modes.

When the system is displaying a mode menu, i.e., requesting a softkey command, there are some additional keys which are always significant. These are the ESCAPE (ESC) key, the ZERO (0) key, the MINUS (-) key, and the COLON (:) key.

The ESC key causes the primary modes described in Section 2.2 to be selected. It is a software warm reset key.

The ZERO key and MINUS keys cause the previous set of softkey options to be selected. They are, in effect, a 'backup menu' command. The COLON key specifies the direct function execution mode as described in Section 2.4.

All target addresses and other data entered in response to prompts are assumed to be in hexadecimal if they are preceded with a "$". Otherwise, they are decimal values.

The softkey concept is a software 'relabeling' of the number keys. At any time when the system is ready for an operator to select a function, it displays a selection for up to nine captioned function keys. Most of the modes do not require all nine options. In that case, the unassigned keys are left blank. If typed, they will be ignored.

The prompt:

Are you sure (Y/N) ?

expects either an upper or lower case "Y" as the affirmative response. Any other entry is considered a negative response.

The following prompts ask for an input or output port number to be entered:

Output to (1/2) ? or Input from (1/2) ?

where: 1 = I/O Port 1
2 = I/O Port 2
CR or SPACE = the console display

If the port is the console display, the user can control the speed of display using the numeric keys 0 through 9, with 0 being the fastest and 9 being the slowest. The SPACE bar stops the display and the pressing of any key resumes the display of data. The ESC key terminates the display and returns to the softkey options.

2.4 DIRECT FUNCTION EXECUTION

Most of the commonly used LCE system functions can be directly called from any softkey menu level using the COLON (:) key. By pressing ":" followed by the software command (e.g., by typing "DUMP" then pressing CR), that function will begin as if it were selected by the standard softkey sequence. Once this mode is entered, it will be maintained until the function being performed uses softkeys for additional prompts, in which case normal softkey execution mode is resumed. Whenever the direct function execution mode is active, the softkey menu prompt line is replaced by a ":" in column 1. The softkey execution mode can also be resumed from this mode by using a ZERO (0) or MINUS (-), which returns to the PRIMARY menu.

A summary of all LCE commands is shown in Figure 2-2 with the direct function execution commands indicated.

```
SYSGEN FUNCTIONS
  BAUD            Select baud and parity options
  WORD LEN        Select serial word length
  DELAY           Set delay after line feed between 0 to 99 ms
  FORMAT          Set record format for data files (";", ":", or Block)
  START CHAR      Set start of file marker character
  STOP CHAR       Set end of file marker character
  SYSTEM          Select the system console port
  TARGET          Select the processor type for emulation
  PROTOTYPE       Select areas to mapped external of LCE target memory
  PROTECT         Set and clear memory areas to be write-protected
  BEEPER          Turn on or off the BELL to indicate errors
  DISPLAY         Show the current SYSGEN setup parameters
  EXIT            Accept the SYSGEN parameters and exit to PRIMARY menu

DEBUG FUNCTIONS
  DUMP            *Show a screen image of user memory (Hex and ASCII)
  MODIFY          *Show user memory and allow changes (Hex and ASCII)
  ASSEMBLER       *Assemble op code text into user memory
  DISASSM         *Disassemble op code image of user memory
  LOAD            *Load a file image into user memory
  SAVE            *Save a file image of user memory
  VERIFY          *Verify a file image with user memory
  STEP            *Execute user program displaying every instruction and
                   associated registers
  RUN             *Execute user program at full speed until any break
                   condition occurs
  SNAPSHOT        *Execute user program displaying only breakpoint
                   instructions and registers
  MATCH           *Execute user program at full speed until a break condition
                   (except BP1 or BP2)
  RESET           *Reset all debug parameters and hardware
  REG             *Show and accept changes to emulator registers
  BREAK           Show and accept changes to the breakpoint conditions

MISCELLANEOUS FUNCTIONS
  LIST            *Show the user symbol table
  CREATE          *Adapt labels changes to the symbols tables
  LOADSYM         *Load a file image of the symbol table
  SAVESYM         *Save a file image of the symbol table
  CLRSYM          *Clear out the present user symbol table
  SYMBOL          *Turn on or off the use of labels in disassembly
  OBJECT          *Turn on or off the creation of object code field in
                   disassembly
  SHOW            *Show status of symbolic options
  SPEED           *Control the rate of display update
  TERMINAL        *Allow transfer between Port 1 and Port 2 directly
  PROM            *LCE PROM Programmer module option
  TRACE           *LCE Trace module option

*Can be directly executed from any menu level using the ":" command.
```

Figure 2-2. LCE Direct Function Execution Commands

SECTION 3

SYSGEN COMMAND MODES

The SYSGEN functions allow the user to define the system configuration for each development application and how the system communicates with other segments of the overall development environment.

All communications between the LCE and the host or peripheral devices are defined by the SYSGEN command. Communications with the host are via the serial interface port. Selection of the SYSGEN mode displays the SYSTEM parameters (Figure 3-1) and captions the softkeys as:

1:PORT 1 2:PORT 2 3:TARGET 4:PROTECT 5:BEEPER 6:DISPLAY 7:EXIT

The SYSGEN functions are, in summary:

PORT 1 - Sets up the I/O characteristics for Port 1 (usually the console terminal). This includes baud rate, word length, line feed delay, file record format, start and stop characters and system console.

PORT 2 - Sets up the I/O characteristics for Port 2 (usually the host computer). This includes baud rate, word length, linefeed delay, file record format, start and stop characters and system console.

TARGET - Defines the processor type and memory mapping range for target memory.

PROTECT - Defines up to four areas of target memory which are write-protected.

BEEPER - Toggles ON/OFF the beep or tone which sounds each time a command is selected.

DISPLAY - Displays the current SYSGEN characteristics.

EXIT - Saves all changes to SYSGEN characteristics and exits to the primary command modes.

The default SYSGEN parameters are displayed (see Figure 2.1a) when the LCE is turned on, following a COLD RESET, or by entering the SYSGEN function from the PRIMARY menu softkey. Pressing a 0 or an ESC key when the menu is displayed exits to the PRIMARY menu but does not update the port related parameters. The SYSGEN function EXIT is the normal way to return to the PRIMARY menu in which case the port parameters are updated.

3.1 PORT

The first two SYSGEN softkeys, PORT 1 and PORT 2, are not functions, but each brings up an additional menu level to configure the interface characteristics of the respective parts.

Exit from the port softkey level to the SYSGEN menu can be done with a menu-backup (0 or -) ESC, or a direct function command (:).

NOTE

Because the port parameters directly affect the console operation, no changes to either port parameters will be made until the SYSGEN is exited by use the EXIT softkey. No other means of exit (i. e., 0, -, ESC or :) will update the port parameters.

When either PORT 1 or PORT 2 softkey is pressed, another softkey menu is displayed:

1:BAUD 2:WORD LEN 3:DELAY 4:FORMAT 5:START CHAR 6:STOP CHAR 7:SYSTEM

3.1.1 BAUD

When the BAUD softkey is pressed, the system responds with an additional softkey menu of different baud rates:

1:110 2:300 3:600 4:1200 5:2400 6:4800 7:9600 8:19200 9:USER

These baud rates can be selected to override the current hardware setting.

If the USER key is pressed, the system prompts for the Control and Command bytes required:

Control=$xx Command=$xx

where: xx is a hex value of data sent to the Control and Command registers in the port ACIA device.

After the baud rate has been selected, the system redisplays the PORT softkey menu.

NOTE

The USER mode requires an understanding of the R6551 ACIA device. This mode is not for general use.

3.1.2 WORD LEN

When the WORD LEN softkey is pressed, the system displays the word length softkey menu:

Serial word length 1:7-BITS 2:8-BITS

This softkey prompt sets up the word length option of 7- or 8-bit words. Press the corresponding softkey to select the proper word length. This option is not meaningful if a USER baud rate has been selected, as it is directly set by the Control register byte. After the word length has been selected, the system redisplays the PORT softkey menu.

3.1.3 DELAY

The DELAY mode selects a delay time of between 0 to 99 milliseconds to be used after the LCE issues any LF ($0A) to the respective port.

When the DELAY key is pressed, the system prompts:

LF delay is xx

where: xx is the default or last entered delay value. The valid range of values is 00 to 99. The default value is 00.

Following the entry and pressing of the CR key, the system redisplays the PORT softkey menu.

3.1.4 FORMAT

The FORMAT mode selects the file record type to be used by the LOAD or SAVE function.

When the FORMAT softkey is pressed, the system responds:

File format is x.

where x is: ; - this is the semicolon record format used by MOS and Rockwell software.
: - this is the colon record format popular on many cross-assemblers.
B - this is the blocked colon format used by the R6500 Cross Assembler for INTEL ISIS.

After the key is pressed, the system redisplays the PORT softkey menu.

3.1.5 START CHAR

The START CHAR mode selects the character which precedes each LOAD or SAVE file.

When the START CHAR softkey is pressed, the system displays:

Start char is $xx

where: xx is a hex value. $FF is no character issued, which is the default.

Following the entry and a CR, the system redisplays the PORT softkey menu.

3.1.6 STOP CHAR

The STOP CHAR mode selects the character which follows each LOAD or SAVE file.

When the STOP CHAR key is pressed, the system responds:

Stop char is $xx

where: xx is a hex value. $FF is no character issued, which is the default.

Following the entry and a CR, the system redisplays the PORT menu.

3.1.7 SYSTEM

Pressing the SYSTEM softkey defines the current port being configured as the system console port. Following the SYSTEM entry the system redisplays the PORT 1 parameter softkeys.

3.2 TARGET

The TARGET softkey defines the processor type and prototype memory range for a debug session. On entry to TARGET, the present parameters are displayed as shown:

```
Target:          65XX is Selected
No Prototype memory
```

or

```
Target           65XX is selected
Prototype Memory
From=0000 To=0000 From=0000 To=0000 From=0000 To=0000 From=0000 To=0000
```

This is followed by the CPU/MPU selection menu.

3.2.1 CPU/MPU Selection

When the TARGET softkey is pressed, the system responds with a new softkey menu:

```
CPU/MPU = 1:65XX  2:6500/1  3:6500/1X  4:6500/4X  5:65CXXX  6:65C00/21
```

This menu allows selection of the different device families which are supported by the debug software.

Press the softkey which corresponds to the connected device family . Once a selection is made, the system redisplays the selection and prompts the user for prototype memory selection.

NOTE

The single target debug ROM (J906-S.DBUG) will not support the R65C02/21 device. This requires an up-grade to be supplied with the R65C00/21 Personality Set.

Ron Collins
714-833-4746

3.2.2 Prototype Memory Selection

After the target CPU/MPU selection, the system responds with the following prompt:

Use Prototype Memory (Y/N)?_ [Y or N]

This question asks if the user is going to map memory onto the prototype circuit or use only target RAM located in the LCE.

The no response tells the system to use only target RAM for emulation memory -- requiring that a Target RAM module (part No. RDC-102X) be in the LCE system.

Press the Y (Yes) key to use prototype memory; the system responds with the following softkey menu:

Set range 1:AREA 1 2:AREA 2 3:AREA 3 4:AREA 4 5:DISPLAY

Any of four areas of prototype memory can be defined by selecting one of the four softkeys marked "AREA x". When an AREA softkey is selected, the system prompts for entry of the address range:

From=$hhhh To=$hhhh

where: hhhh is a 4-byte hexadecimal number or a valid label, followed by a CR. The ranges can be any size, but all mapping is done in 16-byte increments, i. e., if any bytes within that 16-bytes are mapped as prototype, that entire range is prototype.

The SET RANGE menu is redisplayed after the area range is entered.

The DISPLAY softkey will show the present parameters as follows:

Target: 65XX is Selected
Prototype Memory
From=$xxxx To=$xxxx From=$xxxx To=$xxxx From=$xxxx To=$xxxx From=$xxxx To=$xxxx

NOTE

Although the device and prototype memory parameters are kept within SYSGEN on any exit, they are not actually written into the Debug module except on Reset or EXIT from SYSGEN. If a personality set is not attached at these times, the 'Emulator not ready' message will be displayed.

USE "0" TO STORE SETTINGS

3.3 PROTECT

The protect menu functions allow the user to write-protect and unwrite-protect selected areas in memory, unwrite-protect all areas of memory and to display the write-protect status of memory.

When the PROTECT softkey is pressed, the system displays the present protected areas and new softkey menu:

```
No Protected areas
```

or

```
Protected Areas
From=XXXX To=XXXX From=XXXX To=XXXX From=XXXX To=XXXX From=XXXX To=XXXX
```

and

```
1:PROTECT   2:UNPROTECT   3:CLEARALL   4:DISPLAY
```

Use the backup-menu (0 or -) to exit this level.

3.3.1 PROTECT Memory Area

Selecting the PROTECT function allows the user to write protect up to four different address ranges in memory.

When the PROTECT function is selected, the system displays the softkey menu:

```
PROTECT 1:AREA 1  2:AREA 2  3:AREA 3  4:AREA 4  5:DISPLAY
```

Any of four areas of memory can be write-protected by pressing one of the four area softkeys. When an AREA softkey is selected, the system prompts the user for the address range:

```
From=$hhhh   To=$hhhh
```

where: hhhh is a 4-byte hexadecimal number or label.

The ranges can be any size, but all protection is done on 16-byte increments, i. e., if any bytes within a 16-byte range are protected, that entire 16-bytes will be protected.

Use the backup-menu (0 or -) to exit this level.

3.3.2 UNPROTECT Memory Area

Selecting the UNPROTECT softkey allows the user to unprotect any of the four areas previously protected. When the UNPROTECT softkey is pressed, the system displays the softkey menu:

```
UNPROTECT 1:AREA 1  2:AREA 2  3:AREA 3  4:AREA 4  5:DISPLAY
```

Any of the four areas of memory can be unprotected by pressing the appropriate AREA softkey. When an AREA key is pressed, the system requests verification with the prompt:

```
Are You Sure (Y/N)?_ [Y or N]
```

Any response but Y will not affect requested area.

Use the backup-menu (0 or -) to exit this level.

3.3.3 CLEARALL Memory Areas

The CLEARALL softkey removes all write-protection from memory. Because of the impact of this command, an additional prompt must be answered to perform it:

```
Are You Sure (Y/N)?_ [Y or N]
```

Any response but Y will not affect any of the protected areas. This returns to the PROTECT menu.

3.3.4 DISPLAY Protected Memory Areas

When the DISPLAY softkey is pressed, the present protected memory ranges are displayed:

```
No Protected Areas
```

or

```
Protected Areas
FROM=$xxxx TO=$xxxx FROM=$xxxx TO=$xxxx FROM=$xxxx TO=$xxxx FROM=$xxxx TO=$xxxx
```

The system remains in the PROTECT menu after this function.

3.5 BEEPER

The BEEPER softkey simply toggles ON/OFF the issuance of a tone, or beep (a $07), emitted by the system to the console following any error condition. Pressing the BEEPER softkey displays:

```
Beeper is ON
```

or

```
Beeper is OFF
```

3.6 DISPLAY

The SYSGEN DISPLAY function presents the current status of the SYSGEN parameters on the console and redisplays the SYSGEN softkeys. These are the values that will be affected when an EXIT is performed.

3.7 EXIT

The EXIT softkey returns the user to the PRIMARY menu and puts into effect the SYSGEN parameters selected using Port 1, Port 2, Target or Protect command modes. If the EXIT key is not pressed to return to the PRIMARY menu, i.e., pressing the ESC or 0 (zero) key, the parameters selected by the user are not implemented. The system continues to use those previously set.

NOTE

Although the Target and Protect values will be changed in the SYSGEN parameter display on the exit from SYSGEN, they will not be affected (i.e., used by the PMC module) until the EXIT is performed or a WARM RESET occurs.

SECTION 4

DEBUG FUNCTIONS

The debug functions of the LCE allow entry, modification, execution and observation of the user code. For convenience, commands to execute these functions are broken down into two softkey menus:

MEMORY : Examine, change, load or save memory
EMULATE : Execute, step, set breakpoints or registers

Direct movement between these two debug menu 'levels' is accomplished by EMULATE and MEMORY functions on the Memory and Emulation level, respectively.

The MEMORY menu functions allow the user to perform direct operations on the contents of both target and prototype memory. When the MEMORY mode is selected, the system responds by relabeling the softkeys with:

1:DUMP 2:MODIFY 3:ASSEMBLER 4:DISASSM 5:LOAD 6:SAVE 7:VERIFY 8:EMULATE

The MEMORY menu functions are, in summary:

DUMP - Dumps memory data to either an I/O port or to the console.

MODIFY - Modifies one or more bytes in target or prototype memory.

ASSEMBLER - Accepts mnemonic opcodes and symbolic values for entry into target or prototype memory.

DISASSEM - Disassembles target or prototype memory into mnemonic opcodes and symbolic values.

LOAD - Loads target or prototype memory with object code from either I/O port (Port 1 or 2).

SAVE - Creates an object code file from data in target or prototype memory.

VERIFY - Verifies the contents of Target/Prototype memory with an object code file loaded from either I/O port.

EMULATE - Selects the Emulate softkey menu.

The EMULATE menu functions allow the user to directly control the run-time characteristics of target/prototype memory and emulation.

When the EMULATE mode is selected, the system responds by relabeling the softkeys with:

1:STEP 2:RUN 3:SNAPSHOT 4:MATCH 5:RESET 6:REG 7:BREAK 8:MEMORY

The EMULATE menu functions are, in summary:

STEP - Executes the program in target/prototype memory, displaying each instruction as it is executed.

RUN - Executes the program in target/prototype memory at full operational speed and stopping at each breakpoint.

SNAPSHOT - Executes the program in target/prototype memory, displaying each breakpoint but continuing execution.

MATCH - Executes the program in target/prototype memory, outputting a sync signal to the emulator pod at each breakpoint.

RESET - Resets the target CPU to parameters set in the debug software.

REG - Displays and allows changes to the current register values.

BREAK - Displays and allows changes to the software breakpoint.

MEMORY - Selects the MEMORY softkey menu.

4.1 DISPLAY/ALTER MEMORY

4.1.1 DUMP

The DUMP function allows displays data or transfers data in target or prototype memory to an I/O device attached to either port, in a format suitable for viewing (ASCII and hexadecimal).

When the DUMP option is selected, the system displays the following:

From=$[S]hhhh

where: option S selects Single Byte
hhhh is a 4-byte hexadecimal address or a valid label (preceded by a ";"), followed by a CR.

STANDARD DUMP

If only a hexadecimal address or label is entered, then the system prompts the user for the number of 16-byte lines to dump:

Count = dd

where: dd is a decimal value; 00 is continuous. A label preceded by a ";" can also be used.

Following this entry the system prompts the operator for the output device:

Output to (1/2) ?_ [1, 2, CR, or SPACE]

Once the output device has been entered, the system outputs the data in the following format:

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
F400    41 20 73 61 6D 70 6C 65 20 6C 69 6E 65 0D F5 BE  A sample line...
F410    77 68 69 63 68 20 63 61 6E 20 77 72 61 70 20 20  which can wrap
F420    61 72 6F 75 6E 64 20 21 20 3C 3C 3C 3C 3C 3C 3C  around !
```

If the output port is the system console (CR or SPACE), pressing the SPACE bar causes the display to pause; pressing any key causes a paused display to continue. The output display speed can be changed using the 0-9 keys; 9 being the slowest and 0 being the fastest.

After the completion of the dump or the pressing of the ESC key, the system returns to the MEMORY menu.

SINGLE BYTE OPTION

The single byte option is selected by pressing the S key in response to the "From" prompt. The address is then reprompted:

```
Single Byte   From=$hhhh
```

where: hhhh is a 4-byte hex number or a valid label, followed by a CR.

The prompt asks for the address to be displayed continuously. No counter is used but next the output port is requested:

```
Output to (1/2) ?_  [1, 2, CR or SPACE]
```

Once the output port is entered, the system displays the requested location

```
0018  00  .
0018  85  .
0018  07  .
0018  8C  .
0018  11  .
0018  93  (ESC)
```

Pressing the ESC key terminates the function; control returns to the MEMORY menu level.

4.1.2 MODIFY

The MODIFY function allows data in target or prototype memory to be changed, in either hexadecimal or ASCII format.

The system asks for the address of the first byte in memory to modify:

From=$[S,N] hhhh

where: option S selects Single Byte option,
option N selects No Read option,
hhhh is a 4-byte hexadecimal address or a valid label, followed by a CR.

STANDARD MODIFY

If just an address or label is selected, the system displays 16 bytes of data, in hexadecimal format, starting at that address. The data is also displayed in ASCII format to the right of the hexadecimal data in the manner as the DUMP command. The cursor is positioned to the left of the byte to be changed. The user can then:

- Move to the next byte by pressing the SPACE bar. This leaves the skipped data byte in memory unchanged.

- Modify the contents of the current data byte by typing a two-character hexadecimal value.

- Enter ASCII data in the text area by pressing the T key. Note that the hexadecimal value corresponding to the entered ASCII data will not be played until the address is reselected.

- Backup one byte entry, or ASCII character, by pressing the DEL key.

- EXIT the MODIFY function by pressing CR or ESC.

4.2 ASSEMBLY

4.2.1 ASSEMBLER

The ASSEMBLER option invokes a single-pass assembler. The assembler accepts symbolic opcodes entered from either I/O port. The assembled code is stored in target/prototype memory.

The system first asks for the input device:

```
Input from (1/2) ?_  [1, 2, CR or SPACE]
```

Following the entry of the input device type the system then assembles the source code input. Any error messages will be directed to the system console.

The accepted op codes are as described in the R6500 CPU Programming Manual.

Besides the valid opcodes for each target CPU/MPU, there are four additional pseudo-opcodes as described.

Location Counter Definition - "*"

On any new line, the address at which the code is being placed in target/prototype memory can be reassigned with the "*" pseudo-opcode:

```
*=$hhhh
```

where: hhhh is a 4-byte hexadecimal number or a valid label, followed by a CR.

Accumulator Addressing Use A as in LSR .A

Label Definition - ";"

On any new line, a new symbol can be created by using the ";" pseudo-opcode:

;label Value =$hhhh

where: hhhh is a 4-byte hexadecimal number or a valid label (see Section 5), followed by a CR.

NOTE

A label cannot be redefined (i.e., if the "EXISTS=" message is displayed) during the assembly process. If a label is to be redefined, a SYMBOL-CREATE (see Section 5.2) must be performed. This is to prevent accidental changing of label assignments.

Single Byte Entry - ".B"

On any new line, the present location can be initialized to a value by using the .B pseudo-op code (.BYT is also allowed):

.BYT = $hh

where: hh is a 2-byte hexadecimal number or a valid label, followed by a CR.

Additional values can be entered by following entry with a comma. Symbol

NOTE

When the $ prompt is displayed, a symbol can be defined by simply pressing the ; key. The ; character is redisplayed in place of the $ character and a symbol value can be entered.

Exit can be made from any new line by pressing CR twice or ESC once.

NOTE

A label must have a value of less than $FF to be accepted.

Two Byte Entry - ".W"

On any new line, the present location and the address following can be initialized to a value by using the .W pseudo-op code (.WOR is also allowed) as shown:

.WOR = $hhhh

where: hhhh is a 4-byte hexadecimal number or a valid label, followed by a CR.

NOTE

The two bytes accepted are stored in memory low-byte first.

Additional values can be entered by following each entry with a comma.

An example assembly:

```
Addr  Memory      Label   Opcode Operand.....
0200  69 10       START   ADC #$10
0202  65 11               ADC $11
0204  75 12               ADC $12,X
0206  6D 15 14            ADC $1415
0209  7D 17 16            ADC $1617,X
020C  79 19 18            ADC $1819,Y
020F  61 20               ADC ($20),X
0211  71 22               ADC ($22),Y
0213  29 10       ANDING  AND #$10
0215  25 11               AND $11
0217  35 12               AND $12,X
0219  2D 15 14            AND $1415
021C  3D 17 16            AND $1617,X
```

NOTE

Display of the object code (address and memory) is controlled by the SYMBOL trace option in the SYMBOL menu (see Section 5.3). Display of the symbol (label and operand) is controlled by the OBJECT trace option in the SYMBOL menu (see Section 5.4).

4.2.2 DISASSM

The DISASSM function symbolically disassembles object code. The disassembler reads object code from target/prototype memory, converts it to mnemonic code and outputs this information to a user selected output.

Upon selection, the system prompts:

```
From=$hhhh
```

where: hhhh is a 4-byte hexadecimal number or a valid label, followed by a CR.

The entry defines the starting address in target/prototype memory where the disassembly will begin. The system then asks for the number of instructions to be disassembled:

```
Count=dd
```

where: dd is a decimal value; 00 is continuous.

The system then asks for the output device to be used:

```
Output to (1/2) ?_  [1, 2, CR or SPACE]
```

Once the output device has been selected, the system begins outputting the disassembled code. If the output port is the system console, pressing a SPACE key causes the disassembly to pause. A paused output will be resumed upon pressing any key. The output display speed can be changed using the 0-9 keys; 9 being the slowest and 0 being the fastest.

An example disassembly:

```
Addr  Memory      Label   Opcode Operand.....
0200  69 10       START   ADC #$10
0202  65 11               ADC $11
0204  75 12               ADC $12,X
0206  6D 15 144           ADC $1415
0209  7D 17 16            ADC $1617,X
020C  79 19 18            ADC $1819,Y
020F  61 20               ADC ($20,X)
0211  71 22               ADC ($22),Y
0213  29 10       ANDING  AND #$10
0215  25 11               AND $11
0217       ---
```

Following the completion of the disassembly, or the pressing of ESC, the system returns to the MEMORY menu.

4.3 LOAD/SAVE MEMORY

4.3.1 LOAD

The LOAD function loads target/prototype memory with object code from either I/O port. The LOAD option accepts semi-colon, colon, or block colon formats.

Upon selection, the system first prompts:

```
Offset=$hhhh
```

where: hhhh is a 4-byte hexadecimal value or a valid label, followed by a CR.

The offset value is a displacement to the address of the object code file into target/prototype memory.

The system then asks for the input device:

```
Input from (1/2) ? [1, 2, CR or SPACE]
```

NOTE

CR or SPACE defaults to Port 1.

Once the input device is accepted, the LCE issues a start character (if one is assigned) and awaits the object data. A "." is displayed on the console for each record received correctly. A "*" is displayed for each record in error (typically a checksum error or invalid memory). After a transfer of all records or receipt of a stop character (if one is assigned), the system returns to the MEMORY menu.

NOTE

If the object records are sent too fast for the LCE, then a "Port Overrun Error" will result and the Load function will be aborted. If this occurs, try using either a slower baud rate, or adding delays between object records.

4.3.2 SAVE

The SAVE function creates an object code file of data in target/prototype memory and saves it to a user-selected output device. The file is sent in either semicolon or colon format, depending on the SYSGEN parameter selected on the port in use.

Upon selection, the system asks for the address range of target/prototype memory to save followed by the offset.

From=$hhhh To=$hhhh Offset=$hhhh

where: hhhh is a 4-byte hexadecimal value or a valid label, followed by a CR.

NOTE

If the entered "To" address value is lower than the entered "From" address, the ILLEGAL ADDRESS RANGE error message is displayed followed by redisplay of the "From" address prompt.

Following the entry of these parameters, the system asks if there are any other areas of target/prototype memory to be saved:

```
More (Y/N)?_ [Y or N]
```

If Y is pressed, additional prompts for addresses and offset are displayed. If N is pressed, the system then asks for the output port:

```
Output to (1/2) ?_ [1, 2, CR or SPACE]
```

Once the output device is accepted, the LCE issues a start character (if one is assigned) and begins to issue object records. A "." is displayed on the console for each record issued, unless the output device is the system console. After all records are transferred, a stop character is issued (if one is assigned) and the system returns to the MEMORY menu.

4.3.3 VERIFY

The VERIFY function compares a selected object code file with data in target/prototype memory. The VERIFY option accepts semi-colon, colon, or blocked colon formats.

Upon selection, the system first prompts:

```
Offset=$hhhh
```

where: hhhh is a 4-byte hexadecimal value or a valid label, followed by a CR.

The offset value is a displacement added to the address at the object code received for verification in target/prototype memory.

The system then asks for the input device:

```
Input from (1/2) ?_ [1, 2, CR or SPACE]
```

Once the input port is accepted, the LCE issues a start character (if one is assigned) and awaits the object data. A "." is displayed on the console for each record received correctly. A "*" is displayed for each record in error (typically a checksum or byte mismatch). After a transfer of all records or receipt of a stop character (if one is assigned), the system returns to the MEMORY menu.

4.4 EXECUTION

Four modes of execution are selectable: STEP, RUN, SNAPSHOT and MATCH.

4.4.1 STEP

The STEP function executes a program in the target CPU, one instruction at a time, and displays register contents.

Upon selection, the system asks for entry of the Program Counter starting address or a command to start execution at the RESET vector address:

*=hhhh or R

where: hhhh is a 4-byte hexadecimal address or a valid label, followed by a CR.
R specifies RESET vector

The four digit address tells the system from what address in target/prototype memory to begin executing the program. The "R" causes the system to execute a hardware reset, i.e., program execution begins at the address pointed to by the RESET vector. The RESET option requires a verification Y response in order to proceed, otherwise the system returns to the EMULATE menu.

RESET (Y/N) ?_ [Y or N]

Upon a Y response to the RESET verification prompt, or after a valid address is accepted, the system requests the number of instructions to execute:

count = dd

where: dd is a decimal value or a label (preceded by a ";"), 00 is continuous.

Execution begins and continues until either the count is decremented to zero, a break condition (see Section 4.7) is encountered or the ESC key is pressed. Execution will pause when the SPACE bar is pressed and will resume when another key is pressed. The execution display format is:

```
Addr  Memory      Label  Opcode Operand.....  NV-BDIZC AA XX YY SS
F000  A2 AA       TEST   LDX #$AA             NO-00I00 11 AA 33 FE
F002  48                 PHA                  NO-00I00 11 AA 33 FD
F003  EA                 NOP                  NO-00I00 11 AA 33 FD
F004  4C 00 F0           JMP TEST             NO-00I00 11 AA 33 FD
```

The OBJECT and SYMBOL options under the SYMBOL menu affect the results shown during any of the execution function.

NOTE

If ESC is pressed while execution is paused, execution must be resumed before the ESC will take effect.

The valid break conditions for STEP include user quit (ESC), all Breakpoints, External Trigger, Write Protect, Illegal opcode, and Emulator not ready. Termination by a break condition is as shown:

```
Stopped at BP1

Addr  Memory      Label  Opcode Operand.....  NV-BDIZC AA XX YY SS
F002  48                 PHA
F003  EA                 NOP
F004  4C 00 F0           JMP TEST             NO-00I00 11 AA 33 32
```

After termination, control returns to the EMULATE menu.

4.4.2 RUN

The RUN function allows the target CPU to execute at its normal clock speed until either a break condition is detected or the ESC key is pressed. The system prompt for the program counter address as shown:

*=$hhhh or R

where: hhhh is a 4-byte hexadecimal address, or a label, followed by a CR.
R specifies RESET vector

Both of these entries operate in the same manner as described in the STEP function, except no count is requested.

Target CPU execution is initiated at the specified address when CR is pressed. The RUN function then displays the BREAK menu:

1:BP1 2:BP2 3:BP3 4:BP4 5:BP5 6:CLRALL 7:TRACE

where: BP1 through BP5 selects breakpoints 1 through 5. These breakpoints may be set or changed while the target CPU is operating.

CLRALL clears any active breakpoint.

TRACE is a future option.

The valid break conditions for RUN include User quit (ESC), any active breakpoints, External Trigger, Write Protect, Illegal opcode, and Emulator not ready. Termination is as shown for User quit:

```
User quit

Addr  Memory      Label   Opcode Operand..... NV-BDIZC AA XX YY SS
F000  A2 AA TEST          LDX #$AA
F002  48                  PHA                NO-00I00 11 AA 33 F2
F003  EA                  NOP
```

After termination, the system returns to the EMULATE menu.

4.4.3 SNAPSHOT

The SNAPSHOT function performs the same functions as RUN except that when a breakpoint is reached, the system displays the disassembled instruction and register contents then continues execution. Unlike the RUN option, the SNAPSHOT option does not execute in real-time.

The system first prompts for entry of the Program Counter:

*=$hhhh or R

where: hhhh is a 4-byte hexadecimal address, or a label, followed by a CR.
R specifies RESET vector

Both of these entries operate in the same manner as described in the STEP function, except no count is requested.

Target CPU execution is initiated at the specified address when CR is pressed. The SNAPSHOT function then displays the BREAK menu:

1:BP1 2:BP2 3:BP3 4:BP4 5:BP5 6:CLRALL 7:TRACE

where: BP1 through BP5 select breakpoints 1 through 5. The user may set a breakpoint during operation of the SNAPSHOT option.

CLRALL clears any active breakpoint.

TRACE is a future option.

All five breakpoints will pause to display program location and registers then resume operation.

The valid break conditions to stop execution include User quit (ESC), External Trigger, Write protect, illegal opcode, and Emulator not ready. Termination is as shown in RUN description. After termination, the system returns to the EMULATE menu.

4.4.4 MATCH

The MATCH function is identical to the RUN option with one exception. When a breakpoint is detected, on either BP1 or BP2, the system outputs a sync signal through the sync ports located on the emulator pod, but does not stop execution. The system prompts for the program counter as shown:

*=$hhhh or R

where: hhhh is 4-byte hexadecimal address, or a label, followed by a CR.
R specifies RESET vector

Both of these entries operate in the same manner as described in the STEP function, except no count is requested.

Target CPU execution is initiated when CR is pressed. The MATCH function then displays the BREAK menu:

1:BP1 2:BP2 3:BP3 4:BP4 5:BP5 6:CLRALL 7:TRACE

where: BP1 through BP5 select breakpoints 1 through 5. These breakpoints may be set or changed while the target CPU is operating.

CLRALL clears any active breakpoint.

TRACE is a future option.

The two sync outputs, Sync 1 and Sync 2 are used with breakpoints 1 and 2.

The valid break conditions to stop execution include User quit (ESC); breakpoints 3, 4, and 5; External Trigger; Write Protect; illegal opcode; and Emulator not ready. Termination is as shown in RUN description, with the system returning to the EMULATE menu.

4.5 RESET

The RESET function resets the debugger hardware. This includes resetting the target CPU, initializing registers to default values (except PC which becomes the Reset vector value), and re-initializing TARGET and WRITE-PROTECT parameters from SYSGEN into the PMC (similar to EXIT, refer to Section 3.7).

Because of the impact of this command, a verification prompt must be answered the reset will be performed:

```
Are you Sure (Y/N) ?
```

Any response but Y will not affect the operation. The system then returns to the EMULATE menu.

4.6 DISPLAY/ALTER REGISTERS

The REGISTER menu displays the contents of the target CPU registers and allows the data to be changed. Each bit in the status register is also displayed and accessed individually.

When the REG option is selected, the system displays the status of all registers:

```
*=$F000  A=$11  X=$22  Y=$33  P=00-00100  S=$02
```

then displays the REGISTER menu:

```
 1:PC  2:A  3:X  4:Y  5:PS  6:SP  7:ALL  8:KEEP
```

To change a register value, press the softkey corresponding to desired register.

4.6.1 PC (Program Counter)

The PC command displays and then accepts changes to the Program Counter (PC). The form is:

*=$hhhh

where: hhhh is a 4-byte hexadecimal value, or a label, followed by a CR.

4.6.2 A (A Register)

The A command displays and accepts changes to the A register. The form is:

A=$hh

where: hh is a 2-byte hexadecimal value, or a label, followed by a CR.

4.6.3 X (X Register)

The X command displays and accepts changes to the X register. The form is:

X=$hh

where: hh is a 2-byte hexadecimal value, or a label, followed by a CR.

4.6.4 Y (Y Register)

The Y command displays and accepts changes to the Y register. The form is:

Y=$hh

where: hh is a 2-byte hexadecimal value, or a label, followed by a CR.

4.6.5 PS (Processor Status)

The PS command displays and accepts changes to the Processor Status (Status Flag register). Notice that the flag descriptor is shown before the entry as below:

```
NV-BDIZC  P=bb-bbbbb
```

where: b are the binary bits (0 or 1) and SPACE keeps the present value.

4.6.6 SP (Stack Pointer)

The SP command displays and accepts changes to the Stack Pointer. The form is:

```
S=$hh
```

where: hh is a 2-byte hexadecimal value o,r a label, followed by a CR.

4.6.7 ALL

The ALL command displays all register values and allows successive changes to each. The valid entry for each remains the same as described in previous sections. Exit can be made at any time by pressing ESC which allows entries to be kept. The form is:

```
*=$F000  A=$11  X=$22  Y=$33  NV-BDIZC  P=00-00100  S=$FF
```

4.6.8 KEEP

To allow individual access to any registers, and still allow simple error recovery, all changes to register values will be retained only if the REGISTER menu level is exited using the EXIT softkey or if the KEEP function is commanded at the REGISTER menu level. Exit from the REGISTER menu level with 0, -, or ESC will not update any register values.

4.7 BREAK

The BREAK function displays the status of five hardware breakpoints and accepts changes to the status and conditions of a selected breakpoint. The status of a given breakpoint can be either active (execution will break upon satisfied condition) or inactive (no break will occur for the breakpoint regardless of conditions). The break conditions can be any specified combination of address bus (16 lines), data bus (8 lines) or control signal (6 lines) states for the target CPU.

The selection of the BREAK function displays the softkey labels for the breakpoint functions.

```
1:BP1  2:BP2  3:BP3  4:BP4  5:BP5  6:CLRALL
```

A particular breakpoint function can now be selected using the softkeys.

4.7.1 BP1 Through BP5

Pressing one of the breakpoint softkeys (BP1 - BP5) allows the status of the selected breakpoint to be changed. After selecting one, the current status of all five breakpoints is first displayed in the following format:

```
#  Status  Addr Mask Data Mask Re SO Sy RW Nm Ir
1 Inactive hhhh hhhh  hh   hh  N  N  N  N  N  N
2 Active   hhhh hhhh  hh   hh  N  N  N  N  N  N
3 Inactive hhhh hhhh  hh   hh  N  N  N  N  N  N
4 Inactive hhhh hhhh  hh   hh  N  N  N  N  N  N
5 Active   hhhh hhhh  hh   hh  N  N  N  N  N  N
```

where: hhhh is a 4-byte hex value
hh is a 2-byte hex value
N = 0, 1 or X

The breakpoint display fields are defined as follows:

is the breakpoint number.

Status is the active or inactive state of the breakpoint.

Addr is the specified state of the address bus lines that the break will occur, as conditioned by the address mask field (4-digit Mask).

Mask (4-digit) is the mask for the corresponding Addr field. A break will occur only when the state of all the address lines meets the conditions specified for all bits in the Addr and Mask fields. The address mask bit definitions are:

1 = Break only if the address line state corresponding to mask bit position meets the condition specified in the same bit position in the Addr field. A FFFF mask will thus allow break only when all address lines meet the Addr criteria.

0 = Break on any state of the corresponding address line regardless of the condition specified in the same bit position in the Addr field. A 0000 mask will thus cause a break on any address.

Data is the state of the target CPU data bus lines that the break will occur as conditioned by the data mask field (2-digit Mask).

Mask (2-digit) is the mask for the Data field. A break will occur only when the state of all the data lines meets the conditions specified for all bits in the Data and Mask fields. The data mask bit definitions are:

1 = Break only if the data line state corresponding to mask bit position meets the condition specified in the same bit position in the Data field. A FF mask will thus allow break only when all data lines meet the Data criteria.

0 = Break on any state of the corresponding data line regardless of the condition specified in the same bit position in the Data field. A 00 mask will thus cause a break on any state of the data lines.

Re = $\overline{RES}$ line break condition:

X = $\overline{RES}$ line does not affect break.
1 = Break on RES high
0 = Break on RES low

SO = Set Overflow (SO) line break condition

X = SO line does not affect break.
1 = Break on SO high
0 = Break on SO low

Sy = SYNC line break condition

X = SYNC line does not affect break.
1 = Break on SYNC high
0 = Break on SYNC low

RW = R/$\overline{W}$ line break condition
X = R/$\overline{W}$ line does not affect break.
1 = Break on R/W high
0 = Break on R/W low

Nm = $\overline{NMI}$ line break condition
X = $\overline{NMI}$ line does not affect break.
1 = Break on NMI high
0 = Break on NMI low

Ir = $\overline{IRQ}$ line break condition
X = $\overline{IRQ}$ line does not affect break.
1 = Break on IRQ high
0 = Break on IRQ low

The system then displays:

```
Change status (Y/N) ?_ [Y or N]
```

The status of the selected breakpoint number (BP1 through BP5) can now be changed from the keyboard.

If N is pressed, the system returns to the BREAK menu. If Y is pressed, the system toggles the breakpoint status. If the status of the breakpoint is to be changed to inactive, the breakpoint status is changed to inactive, the breakpoint function is ended, and the BREAK softkey menu is displayed.

When the breakpoint is made active, the cursor is positioned over the breakpoint address prompt and the user can begin modifying the breakpoint characteristics. Four digit hexadecimal values are expected for the breakpoint address and address mask value. The bits in the mask value can be thought of as "don't care values". If, for example, the user wants to break at any address in the range $E300 to $E30F, the address value can be any value from $E300 to $E30F and the mask value would be $FFF0. The breakpoint data value and data mask work in the same manner as the address and address mask values. The data mask value must be $FF for the breakpoint to execute properly. All of the remaining values have three value states: 0 for low, 1 for high and X for "don't care".

The breakpoint characteristics other than address are used as modifiers to the address breakpoint condition. Example, if the breakpoint address is $E300, the breakpoint data value is $F4, and both of the mask values are $FFFF, the system breaks when the address is $E300 and the data is $F4.

When changing items in the breakpoint, a CR or SPACE leaves the value unchanged and moves the cursor to the next item in the breakpoint. An ESC terminates the modifying process and leaves the breakpoint unchanged. The menu level is exited by a "0" key to update the new values.

When a breakpoint is encountered in STEP mode, the instruction following the breakpoint is not executed. The display format at break is:

Address - 1	Instruction prior to breakpoint
Address	Instruction breakpoint occurred on
Address	Instruction breakpoint occurred on

Whenever a breakpoint is encountered during the RUN, SNAPSHOT or MATCH mode, the instruction following the breakpoint is executed. The following display at break is:

(Address - 1) (OP)	Instruction prior to breakpoint
Address	Instruction breakpoint occurred on
Address + 1	Instruction executed following breakpoint

The breakpoint options allow flexible address and data triggering. The address of the breakpoint may not appear in the trace at all because an indirect address was created by a memory vector or register index.

NOTE

Whenever the first instruction to execute in RUN or SNAPSHOT mode coincides with a valid breakpoint condition, the previous instruction displayed is address $F401 with disassembly at the user-code at that address. This code was not actually executed, it is merely a remnant of a system execution.

The available breakpoint conditions for the four emulation modes can be summarized as follows:

Break Condition	Breakpoint Mode			
	STEP	RUN	SNAPSHOT	MATCH
User quit (ESC)	X	X	X	X
BP 1 (Sync 1)	X	X	-	-
BP 2 (Sync 2)	X	X	-	-
BP 3, 4, and 5	X	X	-	X
External Trigger	-	X moves stack	X	X
Write Protect	X	X	X	X
Illegal Opcode	X	X	X	X
Emulator Not Ready	X	X	X	X

where: X is an applicable condition for the selected mode.
- is not an available condition for the selected mode.

4.7.2 CLRALL

The CLRALL command makes all breakpoints inactive. Because of the impact of this command, an additional prompt must be answered to perform it:

Are you sure (Y/N) ?

Any response but Y will not affect the present breakpoint status and the system will return to the BREAKPOINT menu.

SECTION 5

SYMBOL MENU FUNCTIONS

The SYMBOL menu functions allow the user to generate and use the symbol table. The symbol table is a list of symbols and their corresponding values. A valid symbol is any combination one to six alphanumeric characters, where the first character is an alphabetic. An existing symbol can be used, or a new symbol defined any time a hex number input is expected - as denoted with the =$ prompt. To enter a symbol (also referred to as a label), type a semicolon (;), followed by the label and a CR. If the label exists, the value is shown for verification as below:

=;LABEL Exists=$0000

where: only CR, DEL or ESC are accepted.

CR accepts the value shown and continues with the function.

DEL indicates that a wrong label value was used and reprompts with "=;" for entry of a new label.

ESC aborts the function, returning to the present menu level.

NOTE

If a label exits, it cannot be reassigned a new value without entering the SYMBOL menu CREATE function (see Section 5.2). This is to prevent accidental changes to symbol table values.

If the label does not exist, a new value is prompted:

=;LABEL Value =$hhhh

where: hhhh is a hexadecimal number, or a valid label, followed by a CR.

Whenever a value is accepted (either from a semi-colon definition or the SYMBOL menu CREATE function), it is alphabetically entered into the symbol table. After the symbol and it's value are entered into the symbol table, the selected function continues.

When the SYMBOL menu is selected, the system responds by displaying the following softkey:

1:LIST 2:CREATE 3:SYMBOL 4:OBJECT 5:LOADSYM 6:SAVE 7:CLRSYM 8:SHOW

The SYMBOL menu functions are, in summary;

LIST - Lists the current contents of the symbol table.

CREATE - Allows the user to enter a symbol and its value into the symbol table or to change the value of a symbol already in the symbol table in the table.

SYMBOL - Toggles the symbolic trace feature on/off.

OBJECT - Toggles the object code display in the disassembler on/off.

LOADSYM - Accepts entries from an input device into the symbol table.

SAVESYM - Creates a symbol table file and saves it to an output device.

CLRSYM - Clears all symbols from the symbol table.

SHOW - Displays the status of the current symbol table.

Same as SaveSym except for space following Symbol and 2 Spaces following Value

5.1 LIST

The LIST function displays the alphabetized contents of the symbol table to the selected output device (typically, the system console as a printer).

When the list function is selected, the system asks for the output port that the contents of the symbol table will be output to:

```
Symbol Table Output to (1/2)?_  [1, 2, CR or SPACE]
```

If there are symbols in the table, they are listed in ascending ASCII code order, with up to six labels per line, for as many lines as required. For labels less than six characters, the symbols are left justified with spaces to the right. A typical table is as shown:

```
LOOP1  0205  LOOP2  0208  NEW    0214  START1 0200  START2 0300  TEST3  0218
```

If no symbols are in the table, the an empty symbol table message is displayed:

```
EMPTY TABLE
```

After the table is completed, the SYMBOL menu is displayed.

5.2 CREATE

The CREATE option allows the user to add or change an entry in the symbol table.

When the CREATE function is selected the label entry prompt is displayed:

```
Label=ssssss
```

where: ssssss is a valid label, entered without a preceding ";", terminated by a CR or SPACE.

The system requires an alphanumeric entry of up to six characters for the symbol name. Once the label is entered, the system checks for an value assigned to the symbol (based on a prior entry). If a value does not exist, the system prompts for entry of the symbol value:

Value =$hhhh

where: hhhh is a hexadecimal number, or ;LABEL, followed by a CR.

If the label has been previously defined, the system displays:

Exists=$hhhh

where: hhhh is a hexadecimal number

and allows entry of a new value for the label. If the user wants to keep the value, simply pressing CR or SPACE keeps the current value.

5.3 SYMBOL

The SYMBOL function toggles the symbolic trace feature on or off.

When the SYMBOL function is selected, the symbol trace is toggled on or off and the resulting state is displayed:

Symbolic Trace is OFF/ON

The default value for this option is ON.

5.4 OBJECT

The OBJECT function toggles on or off the object code column display in the disassembler feature.

When the OBJECT function is selected, the option is toggled on or off and the resulting state is displayed:

Object Code is OFF/ON

The default value for this option is ON.

5.5 LOADSYM

The LOADSYM function allows a symbol table to be loaded from Port 1 or Port 2 into into LCE memory.

When the LOADSYM function is selected, the system prompts for specification of the input port:

Symbol Table Input from (1/2)?_ [1, 2, CR or SPACE]

Once the input device is entered, the LCE issues a start character (if one is assigned) and awaits the object data. An "." is displayed on the console for each 10 characters received. After receipt of a stop character (symbol and 0 must be assigned), the system returns to the SYMBOL menu.

NOTE

As with the MEMORY-LOAD Function, a Port Overrun Error can occur if data is received too fast. For correct symbol usage, the received table must be alphabetized in ascending ASCII code order (as resulting from the SAVESYM function).

5.6 SAVESYM

The SAVESYM function creates a symbol table file and outputs it to a user-selected output device.

When the SAVESYM function is selected, the system asks for specification of the port that the symbol table is to be output:

```
Symbol Table Output to (1/2)?_ [1, 2, CR or SPACE]
```

Once the output device is entered, the LCE issues a start character (if one is assigned) and begins to issue symbol table data. A "." is displayed on the console for each 10 characters, unless the output device is the system console. After all characters are transferred, a stop character is issued (if one is assigned) and the system returns to the SYMBOL menu.

NOTE

The symbol table is output as an alphabetized list, in ASCII code sorted order, of each symbol (6 bytes ASCII, spaces to fill field) and its value (4 bytes ASCII). No inter-field spaces are issued, but a CR is sent every fifth symbol, with the appropriate inter-line delay.

5.7 CLRSYM

The CLRSYM function clears the contents of the current symbol table and returns control to the SYMBOL menu. Because of the impact of this command, an additional prompt must be answered to perform it; any response but Y will not effect the operation. The prompt is:

```
Are You Sure (Y/N)?
```

5.8 SHOW

The SHOW function displays the status of the current symbol table and flags in the following format:

```
(number) Symbols In Table
Symbolic Trace is (ON/OFF)
Object Code is (ON/OFF)
```

SECTION 6

TERMINAL MODE

6.1 TERMINAL

The TERMINAL mode echoes every character entered from the console to the other port. Also, every character received by the console from the other port is sent to the console.

The one character not sent from the console to the other port is CTRL G ($07), which is the character, when typed on the console, terminates the TERMINAL mode.

Press RESET to return to the SYSGEN menu.

SECTION 7

MISCELLANEOUS FUNCTIONS

7.1 SPEED

The SPEED mode allows the user to select the speed at which data is output to the system console. When using the DUMP or DISASSM modes, it is convenient to control how fast the data is displayed on the system console.

When the SPEED mode is selected, the system prompts:

```
Speed= x
```

where: x is a number ranging from 0-9, with 0 being the fastest and 9 the slowest.

NOTE

The speed of the display can also be changed during the DUMP and DISASSM options by simply pressing a numeric key.

7.2 PROM

The PROM function selects the command modes which control the optional LCE PROM Programmer module. Refer to the LCE PROM Programmer User's Manual for operating instructions.

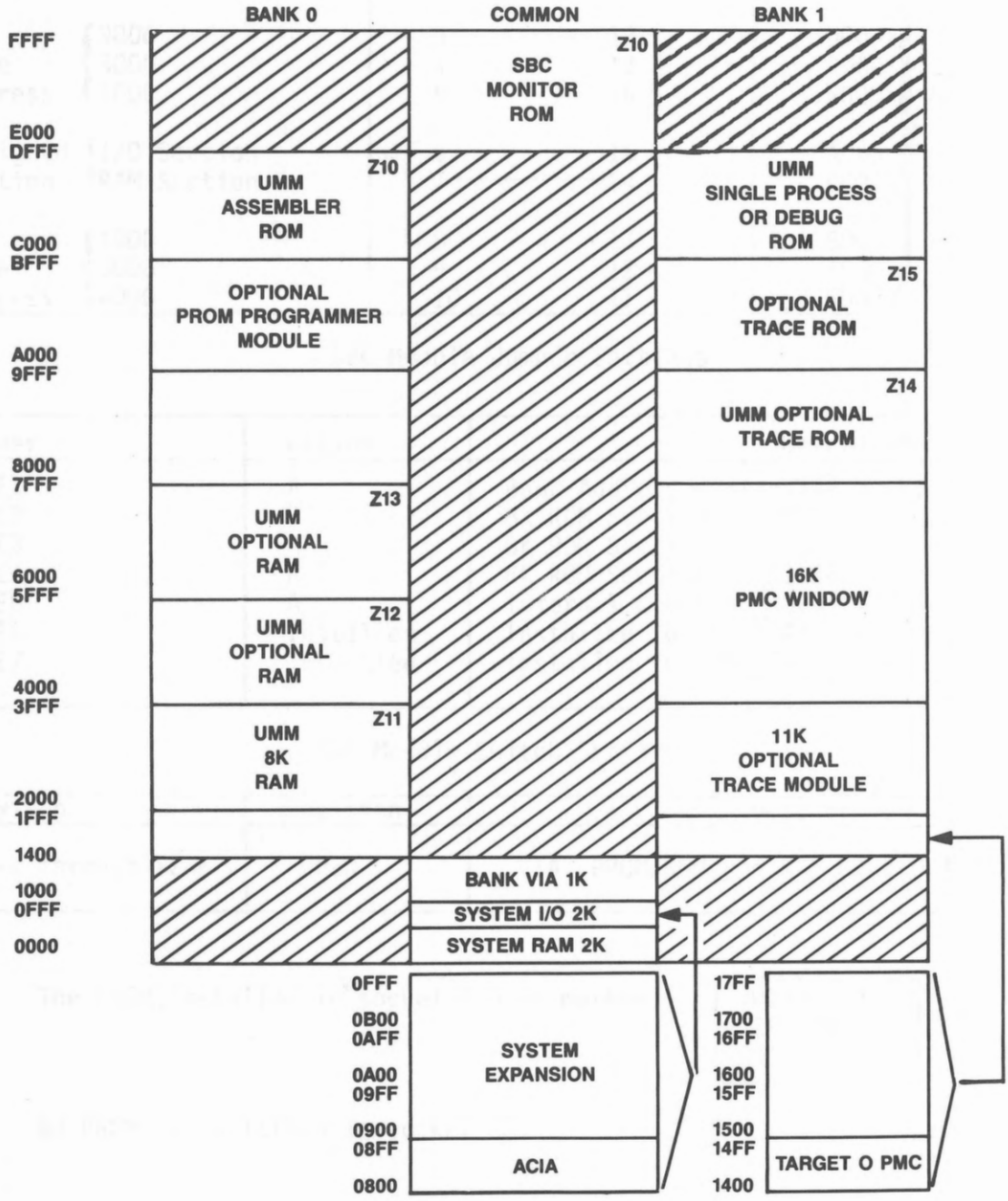
7.3 TRACE

The TRACE function selects the command modes which control the optional TRACE module. Refer to the LCE Trace Module User's Manual for operating instructions.

APPENDIX A

LCE MEMORY MAP

The LCE memory map is made up of two banks, Bank 0 and Bank 1, and a common memory area.



APPENDIX B

LCE JUMPER AND SWITCH SETTINGS

B.1 SBC MODULE

SBC Module Base Address Selection Header

Side 1		Connector Pins		Side 2	
Assigned Section	Z10 PROM Section 1 (Section 0	1 — 20		F000	
		2	19	E000	
Base Address	9000	3	18	D000	Base Address
	3000	4	17	C000	
	2000	5	16	B000	
Assigned Section	I/O Section	6	15	A000	
	RAM Section	7 —	14	0000	
Base Address	1000	8	13	8000	
	3000	9	12	7000	
	4000	10	11	6000	

SBC Module Jumper Settings

Jumper	Setting	Description
E1	A	Bank Address controlled by VIA (PA0)
E2	B	8K ROM Section 0 (2364)
E3	A	8K ROM Section 1
E4	A	8K ROM Section 1 (2364)
E5	A	Internal Clock (1 MHz)
E6	Installed	Installed for Bank Address Control
E7	Installed	Installed for DMA Terminate

SBC Module Switch Settings

Switch	Position	Description
S2-1 through S2-6	OPEN	VIA, PROM/ROM and RAM Bank 0 & Bank 1

The PROM installed in socket Z10 is marked:

```
J904 - 001
LCE MONITOR
SBC Z10
```

No PROM is installed in socket Z8.

B.2 ACIA MODULE

ACIA Module Jumper Setting

Port 2 Data Terminal			Port 1 Data Set		
Jumper	Position	Signal	Jumper	Position	Signal
E1	A	RD	E11	B	RD
E2	A	TD	E12	B	TD
E3	A	RTS	E13	B	RTS
E4	A	CTS	E14	B	CTS
E5	A	DTR	E15	B	DTR
E6	A	DSR	E16	B	DSR
E7	A	DCD	E17	B	DCD
E8	B	CTS not req'd	E18	B	DTR provided
E9	B	DCD not req'd	E19	B	RTS always active
E10	B	DSR not req'd	E20	B	DSR provided

ACIA Module Switch Settings

Switch	Position	Description
S1-1	Open	
S1-2	Open	
S1-3	Open	
S1-4	Closed	
S1-5	Open	Address = $08XX
S1-6	Open	
S1-7	Open	
S1-8	Open	
S1-9	Open	
S1-10	Open	Bank 0 and Bank 1

B.3 UM MODULE

UMM Base Address Selection Header (Z5)

Side 1		Connected Pins		Side 2	
Base Address	0000-1FFF	1	16	Z10	Device Selection
	2000-3FFF	2	15	Z11	
	4000-5FFF	3	14	Z12	
	6000-7FFF	4	13	Z13	
	8000-9FFF	5	12	Z14	
	A000-BFFF	6	11	Z15	
	C000-DFFF	7	10	Z16	
	E000-FFFF	8	9	Z17	

UMM Switch Settings

Switch	Position	Description
S1-1	OPEN	Bank Select
S1-2	OPEN	Bank 0
S1-3	OPEN	Bank 0
S1-4	CLOSED	No Effect
S1-5	CLOSED	
S1-6	CLOSED	
S1-7	OPEN	Write Protect Disabled
S1-8	OPEN	

UMM Jumper Settings

Jumper	Position	Description
E1	A	8K Type Device
E2	A	
E3	B	
E4	B	
E5	B	
E6	B	
E7	A	Chip Select Universal Memory
E8	A	
E9	A	
E10	A	
E11	A	
E12	A	
E13	A	
E14	A	

The ROM installed in socket Z10 is marked:

```
   J905
LCE  ASSEM
UM     Z10
```

The ROM installed in socket Z16 is marked:

```
   J906
LCE  S.DBUG
UM     Z16
```

An 8K-byte RAM device is installed in socket Z11.
No devices are installed in sockets Z12, Z13, Z14, Z15, and Z17.

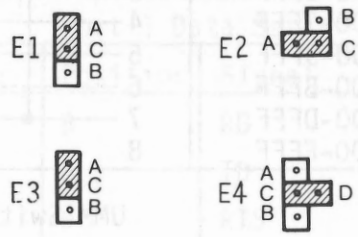
B.4 PMC MODULE

PMC Module Jumpers Settings

Jumpers Configured

E1	E2	E3	E4
A	A	A	C

Jumper Legends



PMC Module Switch Settings

PMC Target ID	Switch 1	2	3	4	5	6	7	8
0	CLOSED	CLOSED	CLOSED	OPEN	CLOSED	OPEN	CLOSED	CLOSED
1*	CLOSED	CLOSED	CLOSED	OPEN	CLOSED	OPEN	CLOSED	OPEN
2*	CLOSED	CLOSED	CLOSED	OPEN	CLOSED	OPEN	OPEN	CLOSED
3*	CLOSED	CLOSED	CLOSED	CLOSED	OPEN	OPEN	OPEN	OPEN

NOTE:
The LCE and RDC use Target ID #0 as the standard ID.
Target ID addresses 1, 2, and 3 are only available on the RDC mainframe and are intended as alternate ID addresses.

APPENDIX C

LCE COMMAND SUMMARY

This section summarizes the LCE commands and defines the command and data entry conventions common to all commands. All items to be input by the user are shown in brackets. If an item is enclosed in parenthesis, only one of the enclosed items is to be input.

COMMAND DEFINITIONS

[SOFTKEY] The selection of any of the function keys are labeled at the bottom of the screen. These are constantly relabeled depending on the functions being used.

[HEX ADDR] A four-character hexadecimal address specifying an address location in memory. The address is terminated by a CR. Errors in the string may be corrected by the use of the BACKSPACE, DELETE, or RUBOUT keys before termination of the string.

[DECIMAL] A two or four digit numeric value specifying a number of instructions. The entry is terminated by a CR. Errors are corrected in the same manner as the [HEX ADDR] definition.

[1 or 2] 1 - I/O Port 1
2 - I/O Port 2

[NAME] A string of from one to six characters specifying the name of a symbol. The string is terminated by a CR or SPACE. Errors in the string may be corrected by use of the BACKSPACE, DELETE or RUBOUT keys before termination of the string.

PRIMARY MENU COMMANDS

```
     1:SYSGEN 2:MEMORY 3:EMULATE 4:SYMBOL 5:TERMINAL 6:PROM 7:TRACE 8:SPEED
```

1:SYSGEN - Displays and sets up the system parameters

```
     1:PORT 1  2:PORT 2  3:TARGET  4:PROTECT  5:BEEPER  6:DISPLAY  7:EXIT
```

2:MEMORY - Displays or alters the contents of target and prototype memory

```
     1:DUMP  2:MODIFY  3:ASSEMBLE  4:DISASSM  5:LOAD  6:SAVE  7:VERIFY  8:EMULATE
```

3:EMULATE - Controls the run-time characteristics of the target CPU emulation.

```
     1:STEP  2:RUN  3:SNAPSHOT  4:MATCH  5:RESET  6:REG  7:BREAK  8:MEMORY
```

4:SYMBOL - Controls the generation and use of the symbol tables

```
     1:LIST  2:CREATE  3:SYMBOL  4:OBJECT  5:LOADSYM  6:SAVESYM  7:CLRSYM  8:SHOW
```

5:TERMINAL - Controls echoing of every character typed on the terminal

6:PROM - Selects the optional LCE PROM Programmer module functions

7:TRACE - Selects the optional LCE Trace module functions

8:SPEED - Defines the speed at which data is output to the console

C.1 SYSGEN MENU COMMAND SUMMARY

1:PORT 1 2:PORT 2 3:TARGET 4:PROTECT 5:BEEPER 6:DISPLAY 7:EXIT

1:PORT 1 Menu and Options

1:BAUD 2:WORD LEN 3:DELAY 4:FORMAT 5:START CHAR 6:STOP CHAR 7:SYSTEM

1:BAUD

1:110 2:300 3:600 4:1200 5:2400 6:4800 7:9600 8:19200 9:USER

1:110 - 110 BPS rate
2:300 - 300 BPS rate
3:600 - 600 BPS rate
4:1200 - 1200 BPS rate
5:2400 - 2400 BPS rate
6:4800 - 4800 BPS rate
7:9600 - 9600 BPS rate
8:19200 - 19200 BPS rate
9:USER - Defines ACIA command and control register contents
Control=$(HEX VALUE) Command=$(HEX VALUE)

2:WORD LEN

Serial word length 1:7 BITS 2:8 BITS

1:7 BITS - Word length of 7 bits
2:8 BITS - Word length of 8 bits

3:DELAY

LF delay is [NUMBER] - Line Feed (LF) delay is (0 through 99) ms

4:FORMAT

File format is [; or : or B] - Format character for each record

5:START CHAR

Start char is $(HEX VALUE or FF for none) - Start file character

6:STOP CHAR

Stop char is $(HEX VALUE or FF for none) - Stop file character

7:SYSTEM

Specifies current port as system console port; no displayed parameters

2:PORT 2 Menu and Options

Same as Port 1 Options

3:TARGET Menu and Options

Target: 65XX is Selected (example)
No Prototype Memory (example)
CPU/MPU = 1:65XX 2:6500/1 3:6500/1X 4:6500/4X 5:65CXXX 6:65C00/21

1:6500 - Select R6500 CPU

Use Prototype Memory (Y/N)? [Y,N]
Set range 1:AREA 1 2:AREA 2 3:AREA 3 4:AREA 4 5:DISPLAY (if Y)
From=$[HEX ADDR] To=$[HEX ADDR] (for each area)

2:6500/1 - Select R6500/1 MCU

Use Prototype Memory (Y/N)? [Y,N]
Set range 1:AREA 1 2:AREA 2 3:AREA 3 4:AREA 4 5:DISPLAY (if Y)
From=$[HEX ADDR] To=$[HEX ADDR] (for each area)

3:6500/1X - Select R6500/1X family of MCUs

```
Use Prototype Memory (Y/N)? [Y,N]
Set range   1:AREA 1  2:AREA 2  3:AREA 3  4:AREA 4  5:DISPLAY (if Y)
From=$[HEX ADDR]  To=$[HEX ADDR] (for each area)
```

4:6500/4X - Select R6500/4X family of MCUs

```
Use Prototype Memory (Y/N)? [Y,N]
Set range   1:AREA 1  2:AREA 2  3:AREA 3  4:AREA 4  5:DISPLAY (if Y)
From=$[HEX ADDR]  To=$[HEX ADDR] (for each area)
```

5:65CXXX - Select R65CXXX family of CPUs

```
Use Prototype Memory (Y/N)? [Y,N]
Set range   1:AREA 1  2:AREA 2  3:AREA 3  4:AREA 4  5:DISPLAY (if Y)
From=$[HEX ADDR]  To=$[HEX ADDR] (for each area)
```

6:65C00/21 - Select R65C00/21 MCU

```
Use Prototype Memory (Y/N)? [Y,N]
Set range   1:AREA 1  2:AREA 2  3:AREA 3  4:AREA 4  5:DISPLAY (if Y)
From=$[HEX ADDR]  To=$[HEX ADDR] (for each area)
```

4:PROTECT Menu and Options

Protected Areas (example)
From=1000 To=1FFF From=0000 To=0000 From=0000 To=0000 From=0000 To=0000

1:PROTECT 2:UNPROTECT 3:CLEARALL 4:DISPLAY

1:PROTECT - Write protects areas of memory

PROTECT 1:AREA 1 2:AREA 2 3:AREA 3 4:AREA 4 5:DISPLAY
From=$[HEX ADDR] To=$[HEX ADDR]

2:UNPROTECT - Remove write-protection from areas from memory

UNPROTECT 1:AREA 1 2:AREA 2 3:AREA 3 4:AREA 4 5:DISPLAY
Are You Sure (Y/N)? (Y,N)

3:CLEARALL - Removes write-protection from all areas of memory

Are You Sure (Y/N)? (Y,N)

4:DISPLAY - Displays write-protected areas of memory

No Protected Areas
or
Protected Areas (example)
From=1000 To=1FFF From=0000 To=0000 From=0000 To=0000 From=0000 To=0000

5:BEEPER

Beeper is ON/OFF

6:DISPLAY - Display current SYSGEN parameters (Example)

```
        Low Cost Emulator       Version 1.0
      Rockwell International   copyright 1984

     Default switches :   9600 baud   8 bits    Warm Reset

     Port 1:   9600 baud   8 bits     [SYSTEM PORT]   Beeper is OFF
     LF delay is 00 ms   File format is ;    Start char is FF   Stop char is FF

     Port 2:   9600 baud   8 bits
     LF delay is 00 ms   File format is ;    Start char is FF   Stop char is FF

     Target:     65XX is Selected
     No Prototype Memory
     No Protected Areas

     *** SYSTEM OPTIONS ***

      8K of System RAM
     Prom Programmer option is not present
     Trace option is not present

      1:PORT 1  2:PORT 2  3:TARGET  4:PROTECT  5:BEEPER  6:DISPLAY  7:EXIT
```

7:EXIT - Resets SYSGEN parameters to currently selected values

No display.

C.2 MEMORY MENU COMMAND SUMMARY

```
     1:DUMP  2:MODIFY  3:ASSEMBLE  4:DISASSM  5:LOAD  6:SAVE  7:VERIFY

1:DUMP       - Dump selected area of memory to I/O port

     From=$[HEX ADDR]  Count=[DECIMAL]
    Output to (1/2) ? [1, 2, CR, or SPACE]

2:MODIFY     - Modify memory data

     From=$[HEX ADDR]

3:ASSEMBLE   - Execute assembler

     Input from (1/2) ? [1, 2, CR, or SPACE]

4:DISASSM    - Disassemble object code

     From=$[HEX ADDR]  Count=[DECIMAL]
    Output to (1/2) ? [1, 2, CR, or SPACE]

5:LOAD       - Load memory with object code

    Offset=$[HEX ADDR]
    Input from (1/2) ? [1, 2, CR, or SPACE]

6:SAVE       - Save memory data to object code file

     From=$[HEX ADDR]   To=$[HEX ADDR]  Offset=$[HEX ADDR]  More? (Y,N)
      Output to (1/2) ? [1, 2, CR, or SPACE]

7:VERIFY     - Verify target memory

     Offset=$[HEX ADDR]
     Input from (1/2) ? [1, 2, CR, or SPACE]
```

C.3 EMULATE MENU COMMAND SUMMARY

```
     1:STEP  2:RUN  3:SNAPSHOT  4:MATCH  5:RESET  6:REG  7:BREAK  8:MEMORY

1:STEP    - Execute program inSTEP mode

     *=$[HEX ADDR] or R  Count=[DECIMAL] (if R not pressed)
     *=Reset (Y/N)?  (if R pressed)
      Count=[DECIMAL]

2:RUN     - Execute program in RUN mode

     *=$[HEX ADDR] or R
     *=Reset (Y/N)?  (if R pressed)

      1:BP1  2:BP2  3:BP3  4:BP4  5:BP5  6:CLRALL 7:TRACE

3:SNAPSHOT- Execute program in SNAPSHOT mode

     *=$[HEX ADDR] or R
     *=Reset (Y/N)?  (if R pressed)

      1:BP1  2:BP2  3:BP3  4:BP4  5:BP5  6:CLRALL 7:TRACE

4:MATCH   - Execute program in MATCH mode

     *=$[HEX ADDR] or R
     *=Reset (Y/N)?  (if R pressed)

      1:BP1  2:BP2  3:BP3  4:BP4  5:BP5  6:CLRALL 7:TRACE

5:RESET   - Reset target CPU

     Are you sure (Y/N)? [Y or N]
```

6:REG - Display/Alter Registers

1:PC 2:A 3:X 4:Y 5:PS 6:SP 7:ALL 8:KEEP

1:PC - Select Program Counter

*=$[HEX ADDR]

2:A - Select Accumulator register

A=$(HEX VALUE)

3:X - Select X register

X=$(HEX VALUE)

4:Y - Select Y register

Y=$(HEX VALUE)

5:PS - Select Processor Status register

P=(BINARY)

6:SP - Select Stack Pointer

S=$(HEX VALUE)

7:ALL - Select all registers for change

*=$hhhh A=$hh X=$hh Y=$hh NV-BDIZC P=bb-bbbbb S=$hh

8:KEEP - Write new entries to register

No display

```
7:BREAK   - Select and define breakpoints

     1:BP1  2:BP2  3:BP3  4:BP4  5:BP5  6:CLRALL

     #  Status  Addr Mask Data Mask Re SO Sy RW Nm Ir
     1 Inactive hhhh hhhh  hh   hh  N  N  N  N  N  N
     2 Active   hhhh hhhh  hh   hh  N  N  N  N  N  N
     3 Inactive hhhh hhhh  hh   hh  N  N  N  N  N  N
     4 Inactive hhhh hhhh  hh   hh  N  N  N  N  N  N
     5 Active   hhhh hhhh  hh   hh  N  N  N  N  N  N
      Change status? (Y/N) [Y or N]
```

C.4 SYMBOL MENU COMMAND SUMMARY

```
     XXXX Symbols In Table
     Symbolic Trace is ON/OFF
     Object Code is ON/OFF

     1:LIST  2:CREATE  3:SYMBOL  4:OBJECT  5:LOADSYM  6:SAVESYM  7:CLRSYM  8:SHOW

1:LIST    - List the current symbol table

     Symbol Table Output to (1/2) ? [1, 2]

2:CREATE  - Create new symbol table entry

     Label=[NAME]    Value=$[HEX VALUE]

3:SYMBOL  - Toggle trace on/off

     Symbolic Trace is ON/OFF

4:OBJECT  - Toggle disassembly object code column off/on

     Object Code is ON/FF

5:LOADSYM - Load new entry in symbol table

     Symbol Table Input from (1/2) ? [1, 2, CR or SPACE]
     Input from (1/2) ? [1, 2, CR or SPACE]

6:SAVESYM - Save symbol table
     Symbol Table Output to (1/2) ? [1, 2, CR, or SPACE]

7:CLRSYM  - Clear symbol table

     Are You Sure (Y/N) ? [Y or N]
```

8:SHOW - Display status of symbol table

XXXX Symbols In Table
Symbolic Trace is ON/OFF
Object Code is ON/OFF

C.5 TERMINAL COMMAND

Terminal - Echo characters from console to host

C.6 PROM COMMANDS

See LCE PROM Programmer Module User's Manual

C.7 TRACE COMMANDS

See LCE Trace Modules User's Manual

C.8 SPEED COMMAND - Set speed of output to console

Speed=(0 through 9)

APPENDIX D

PERSONALITY SET ERROR MESSAGES

This appendix provides a list of error messages which are displayed in the event of an error occurring during debug while using an R65O2-R65CO2 or R65OO/* Personality Set.

Emulator not ready

The emulator device is not functioning properly. The emulator device could be powered down (as in the case when using prototype power for the emulator). The PMC module could be improperly configured, or the PMC could be disconnected from the personality set emulator pod module housing. Another possible cause for this error is that the CPU locked up attempting to execute an illegal instruction, in which case the emulator should be reset before continuing. Also, if the READY line is low this error may appear (the READY line must be high when using the emulator for the system routines, e. g., memory dump, to run).

Memory failure at address XXXX, is XX should be XX

The system could not write to target or prototype memory properly. This error will also occur if a modify command is attempted at a write protected address.

Illegal op code error

The emulator CPU tried to execute an op code that is not a valid member of its instruction set. The message indicates that the emulator CPU has been halted and displays the instruction before the illegal op code, the address of the illegal op code, and the instruction executed after the illegal op code.

Write protect error

See Section 3 of the LCE System User's Manual or Section 11 (Memory Control) of the RDC System User's Manual (as applicable) for an explanation of user protected memory procedures.

APPENDIX E

PROGRAM ASSEMBLY AND DOWNLOAD ON THE IBM PERSONAL COMPUTER

This appendix describes how to assemble a program, written in 6500 assembly language for execution by an R65XX, R65CXX or R6500/* CPU, on the IBM Personal Computer (or equivalent) and how to download the generated object code to the LCE.

SOURCE CODE GENERATION

There are many word processors and/or screen editors available that can be used to input and edit the source code on the IBM PC. However, for this appendix, WORDSTAR is assumed. When using WORDSTAR, be sure to create a Non-document file because a document file contains extended ASCII characters which are not accepted by the assembler. Once the source code is entered and saved on disk, the program can be assembled.

PROGRAM ASSEMBLY

Since the R65C02 and R6500/* use the R6502 extended instruction set, an assembler is needed that assembles the expanded instruction set. There are two cross-assemblers now available that assemble the extended 6500 instructions. They are:

1. 2500 AD Software
2. AVOCET Systems Inc.

The cross assembler from 2500 AD Software is a relocatable macro assembler with a linking loader. The object code is generated in Intel record format (colon format) only.

The cross assembler from AVOCET Systems Inc. is an absolute assembler which can create both Intel record format and Rockwell's semi-colon format.

These two cross-assemblers are very similar and either one can assemble R65XX, R65CXX and R6500/* code. One difference between the two assemblers is that since the 2500 AD Software assembler is a macro assembler, the assemble time to generate object code for the same program is usually longer.

Follow the instructions in the appropriate user's manual to assemble the program. Once the generate object code is saved on disk, this code is ready for the download to the LCE.

OBJECT CODE DOWNLOAD

Download the object code from serial port COM1 on the IBM PC to PORT 2 on the LCE.

1. Be sure that the IBM PC serial port COM1 is connected to LCE PORT 2.

2. Configure PORT 2 on the LCE:

   a. Return to PRIMARY menu.
   b. Select SYSGEN to display SYSGEN functions.
   c. Select PORT 2 to display PORT 2 parameters.
   d. Select BAUD to display PORT 2 baud rates.
   e. Select 2400 to select 2400 baud.
   f. Press the 0 key then the 7 key to return to the PRIMARY menu.

3. Configure COM1 on the IBM PC.

   a. Return to DOS command level.
   b. Select 2400 baud, 8 data bits and 1 stop bit using the MODE command:

      MODE COM1:24,N,8,1,P

4. Initiate the LOAD function on the LCE.

   a. Return to the MEMORY menu.
   b. Select the LOAD function
   c. Select 0 offset
   d. Select input from PORT 2.

5. Initiate download on the IBM PC.

   Copy the object code file to COM1:

   ```
   COPY FILENAME.XXX COM1:
   ```

6. At the completion of the download, the LCE displays to the MEMORY menu.

APPENDIX F

PROGRAM ASSEMBLY AND DOWNLOAD ON THE INTEL DEVELOPMENT SYSTEM

This appendix describes how to assemble a program, written in 6500 assembly language for execution by an R65XX, R65CXX or R6500/* CPU, on an Intel Development System and how to download the generated object code to the LCE.

SOURCE CODE GENERATION

Enter and edit the source program using the development system's text editor. Save the source program on disk when editing is complete.

PROGRAM ASSEMBLY

There are two Rockwell R6500 cross assemblers available for use on Intel development systems. One version (part no. RDC-2000) is provided on 8-inch floppy disk for installation and operation on an Intel Development System. The other version (part no. RDC-2005) is supplied on 5¼-inch minifloppy disk for installation and operation on an Intel Personal Development System (PDS). Both assemblers generate object code for the extended instructions available on the R65C02 and R6500/* CPUs. Assembler operation is similar and is described in the R6500 Cross Assembler for Intel Development System User's Manual.

Follow the instructions in the user's manual to assemble the program. Once the generated object code is saved on disk, this code is ready for the download to the LCE.

OBJECT CODE DOWNLOAD

Download the object code from the development system to PORT 2 on the LCE.

1. Be sure that the development system output port is connected to LCE PORT 2.

2. Configure PORT 2 on the LCE:

   a. Return to PRIMARY menu.
   b. Select SYSGEN to display SYSGEN functions.
   c. Select PORT 2 to display PORT 2 parameters.
   d. Select BAUD to display PORT 2 baud rates.
   e. Select 2400 to select 2400 baud.
   f. Press the 0 key then the 7 key to return to the PRIMARY menu.

3. Configure the serial channel output on the development system.

   One source of information is appendices A and B in the Intellec* Series II Model 22X/23X Installation Manual (Intel Order No. 9800559-06).

4. Setup the LCE Port 2 to expect ":" format object code.

   a. Return to SYSGEN menu.
   b. Select PORT 2.
   c. Select FORMAT.
   d. Select ":".

5. Initiate the LOAD function on the LCE.

   a. Return to the MEMORY menu.
   b. Select the LOAD function.
   c. Select 0 offset.
   d. Select input from PORT 2.

6. Initiate download on the Intel Development System.

   a. Convert the generated object file to hexadecimal using "OBJHEX".
   b. COPY FILENAM.HEX :TO:

7. At the completion of the download, the LCE displays to the MEMORY menu.

* Intellec is a registered trademark of Intel.

SEMICONDUCTOR PRODUCTS DIVISION REGIONAL ROCKWELL SALES OFFICES

HOME OFFICE

Semiconductor Products Division
Rockwell International
4311 Jamboree Road
P.O Box C, MS 501-300
Newport Beach, California
92658-8902
(714) 833-4700
TWX: 910 591-1698

UNITED STATES

Semiconductor Products Division
Rockwell International
1842 Reynolds
Irvine, California 92714
(714) 833-4655
TWX: 910 595-2518

Semiconductor Products Division
Rockwell International
3375 Scott Blvd., Suite 410
Santa Clara, California 95054
(408) 980-1900
TLX: 756560

Semiconductor Products Division
Rockwell International
2001 N. Collins Blvd., Suite 103
Richardson, Texas 75080
(214) 996-6500
TLX: 73-307

Semiconductor Products Division
Rockwell International
10700 West Higgins Rd., Suite 102
Rosemont, Illinois 60018
(312) 297-8862
TWX: 910 233-0179 (RI MED ROSM)

Semiconductor Products Division
Rockwell International
5001B Greentree
Executive Campus, Rt. 73
Marlton, New Jersey 08053
(609) 596-0090
TWX: 710 940-1377

FAR EAST

Semiconductor Products Division
Rockwell International Overseas Corp.
Itohpia Hirakawa-cho Bldg.
7-6, 2-chome, Hirakawa-cho
Chiyoda-ku, Tokyo 102, Japan
(03) 265-8806
TLX: J22198

Rockwell Collins International
Tai Sang Commercial Bldg., 11th Floor
24-34 Hennessy Rd.
Hong Kong
(5) 274-321
TLX: 74071 HK

EUROPE

Semiconductor Products Division
Rockwell International GmbH
Fraunhoferstrasse 11B
D-8033 Munchen-Martinsried
West Germany
(089) 857-6016
TLX: 0521/2650 rimd d

Semiconductor Products Division
Rockwell International Limited
Heathrow House, Bath Rd.
Cranford, Hounslow,
Middlesex, TW5 9QW England
(01) 759-2366
TLX: 851-25463

Semiconductor Products
Rockwell Collins Italiana S.P.A.
Via Boccaccio, 23
20123 Milano, Italy
(02) 498-7479
TLX: 316562 RCIMIL 1

Semiconductor Products Division
Rockwell International
Immeuble de Bureaux Evry 2
523, Place des Terasses
91034 Evry Cedex, France
078.28.88

YOUR LOCAL REPRESENTATIVE

11/84